

ABSTRACT

Title of Dissertation: **EMERGENT BEHAVIORS IN ADAPTIVE
DYNAMICAL NETWORKS WITH
APPLICATIONS TO BIOLOGICAL AND
SOCIAL SYSTEMS**

Brandon Marc Alexander
Doctor of Philosophy, 2021

Dissertation Directed by: **Professor Michelle Girvan
Department of Physics**

In this thesis, we consider three network-based systems, focusing on emergent behaviors resulting from adaptive dynamical features.

In our first investigation, we create a model for gene regulatory networks in which the network topology evolves over time to avoid instability of the regulatory dynamics. We consider and compare different rules of competitive edge addition that use topological and dynamical information from the network to determine how new network links are added. We find that aiming to keep connected components small is more effective at preventing widespread network failure than limiting the connections of genes with high sensitivity (i.e., potential for high variability across conditions). Finally, we compare our results to real data from several species and find a trend toward disassortativity over evolutionary time that is similar to our model for structure-based selection.

In our second investigation, we introduce a bidirectional coupling between a phase synchronization model and a cascade model to produce our ‘sync-contagion’ model. The

sync-contagion model is well-suited to describe a system in which a contagious signal alerts individuals to realign their orientations, where ‘orientation’ can be in the literal sense (such as a school of fish escaping the threat of a predator) or a more abstract sense (such as a ‘political orientation’ that changes in response to a hot topic). We find that success in realigning the population towards some desired target orientation depends on the relative strengths of contagion spread and synchronization coupling.

In our third and final investigation, we attempt to forecast the complex infection dynamics of the COVID-19 pandemic through a data-driven reservoir computing approach. We focus our attention on forecasting case numbers in the United States at the national and state levels. Despite producing adequate short-term predictions, we find that a simple reservoir computing approach does not perform significantly better than a linear extrapolation. The biggest challenge is the lack of data quantity normally required for machine learning success. We discuss methods to augment our limited data, such as through a ‘library-based’ method or a hybrid modeling approach.

EMERGENT BEHAVIORS IN ADAPTIVE DYNAMICAL
NETWORKS WITH APPLICATIONS TO BIOLOGICAL
AND SOCIAL SYSTEMS

by

Brandon Marc Alexander

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2021

Advisory Committee:
Professor Michelle Girvan, Chair
Professor Brian Hunt
Professor Edward Ott
Professor Derek Paley
Professor Bill Fagan

© Copyright by
Brandon Marc Alexander
2021

Acknowledgments

First and foremost, I would like to thank my committee, especially my advisor Prof. Michelle Girvan. Without her knowledge and guidance, this research and my graduate school career would not have been possible.

Second, I would like to thank my family and loved ones. It was my mother's dream before it was mine that I would some day earn a PhD. She raised me into who I am today and deserves the majority of the credit for getting me this far. This work would not have been possible without the constant love and support of my mother Cathy, my partner Dominick, my sister Jaclynn, my step-father John, my father Carman, and my grandparents Ken and Patricia.

Next, I would like to thank my colleagues. My office-mates Graham and Sarah helped pass many hours with random discussions, giving necessary distractions for our overworked brains. We went through this journey together, and I look forward to seeing them too reach this milestone.

Finally, I would like to thank my close friends. Although we haven't been able to see each other for over a year, we've been able to enjoy countless nights online. Thank you to Abbey, Chip, Collin, Dan, Forrest, Hannah, Joe, Josh, Nick, Nikki, Robert, Scott, Shawn, and Vigg.

Table of Contents

Preface	ii
Acknowledgements	ii
Table of Contents	iii
List of Tables	v
List of Figures	vi
List of Abbreviations	xiv
Chapter 1: Introduction	1
1.1 Phase transitions and assortativity in models of gene regulatory networks evolved under different selection processes	3
1.2 The sync-contagion model of dual information spread	5
1.3 A reservoir computing approach to forecasting COVID-19 data	7
Chapter 2: Phase transitions and assortativity in models of gene regulatory networks evolved under different selection processes	10
2.1 Introduction	10
2.2 Methods	14
2.2.1 Modeling failure propagation in gene networks as a percolation problem	14
2.2.2 Tracking network components	17
2.2.3 Network growth via selection	18
2.3 Results	20
2.3.1 Visualizing networks evolved via different selection rules	20
2.3.2 Growth of giant components	23
2.3.3 Maximizing yield	26
2.3.4 Correlations	28
2.3.5 Connecting with data	31
2.4 Discussion	33
Chapter 3: The sync-contagion model of dual information spread	36
3.1 Introduction	36
3.2 Model and methods	39

3.2.1	Sync model	39
3.2.2	Contagion model	41
3.2.3	Combining the models: the sync-contagion model	43
3.2.4	Applications	47
3.2.5	Parameter restrictions	48
3.2.6	Measures	49
3.3	Results	52
3.3.1	General behavior	52
3.3.2	Modifying the width σ of the noise distribution	53
3.3.3	Modifying the coupling strength κ	56
3.3.4	Modifying the transmission coefficient β	59
3.3.5	Behavior classification	61
3.4	Discussion	62
Chapter 4: A reservoir computing approach to forecasting COVID-19 data		66
4.1	Introduction	66
4.2	Methods	68
4.2.1	Data preprocessing	68
4.2.2	Reservoir computer architecture	71
4.2.3	Recursive least squares	76
4.2.4	Baseline predictors	76
4.2.5	Library-augmented reservoir computing	77
4.2.6	Predictor evaluation	78
4.3	Results	80
4.3.1	Recursive least squares and forgetting factor selection	81
4.3.2	Predictor results	82
4.3.3	Library-augmented reservoir computing applied to U.S. COVID-19 case data	92
4.3.4	Autoencoder embedding of state-level COVID-19 data	95
4.4	Discussion	100
Appendix A: Supplementary material for Chapter 2		102
A.1	Locally treelike features	102
A.2	Selection with $m = N$	103
Appendix B: Complications with data preprocessing of COVID-19 case data		105
B.1	Negative number of cases reported	105
B.2	Irregular reporting	106
B.3	Zero cases reported	106
B.4	Invertibility	107
Bibliography		109

List of Tables

2.1	Number of nodes, edges, and the in-out assortativity for the <i>E. coli</i> , yeast, fruit fly, mouse, and human transcriptional networks. These in-out assortativity values are also the final values in Fig. 2.6	34
4.1	List of hyperparameter values used in training the reservoir computers, unless otherwise noted.	81

List of Figures

- 2.1 Structure of networks formed under our four different selection rules on a network of size $N = 100$, stopped after $E/N = 1.75$. Nodes are sorted according to sensitivity with least sensitive at the top of the circle and increasing in sensitivity clockwise. If a node is damaged, it is filled in blue. Nodes in GOUT are enlarged and filled in black (if undamaged) or blue (if damaged); all other nodes are filled in white. Isolated nodes with no links are placed below each network with their corresponding locations in the circular layout left empty. Thick links connect nodes in GOUT. Links are blue if they connect two damaged nodes, black if they connect nodes in GOUT that are not both damaged, or gray otherwise. We see that GOUT is much larger for the random and sensitivity-based processes as compared with the structure-based and hybrid processes. Both the sensitivity-based and hybrid processes show a higher density of links between low sensitivity nodes, which is much more pronounced in the former. 21
- 2.2 Network structure for a network of size $N = 100$ grown under the four selection rules stopped after $E/N = 5.75$. As in Fig. 2.1, nodes are sorted according to sensitivity, with least sensitive at the top of the circle and increasing in sensitivity clockwise and with isolated nodes left out of the circular layout and placed below each network. We now focus on GOUT* rather than GOUT. Nodes belonging to GOUT* are enlarged and red with thick red links connecting between them. Damaged nodes that do not belong to GOUT* are blue, with blue links connecting pairs of damaged nodes. All other nodes are colored white, and all other links are colored gray. Even though the size of GOUT for the sensitivity-based process is similar to the random process, the former has a much smaller GOUT*. The structure-based and hybrid processes show even smaller GOUT*, despite a large number of connections between damaged nodes. . 22

2.3	Growth of the connected components on the full network (left) and the damaged subnetwork (right). We compare growth under random selection to our three competitive rules. Results shown are the averages of 100 runs on networks of size $N = 10^5$. We omit GIN and GIN* due to their symmetry with GOUT and GOUT*, respectively. Of the selection rules considered, we see that structure-based selection (blue curves) does best at delaying the phase transition for the full network, while the structure-based and hybrid selection (red curves) exhibit simultaneous transitions when we consider only the damaged subnetwork, with the hybrid selection showing greater suppression of the component size after the transition.	24
2.4	The average yield of networks grown under our four selection rules, as described by Eq. (2.5). Results are averaged over 100 runs on networks of size $N = 10^5$. Hybrid and structure-based selection processes show similar yield profiles, consistently outperforming both random and sensitivity-based yields.	27
2.5	In-out assortativity (top) and the degree-sensitivity correlation (bottom) for our four selection rules. Results shown are averages over 100 runs on networks of size $N = 10^5$. While the random and sensitivity-based process are assortatively neutral (i.e., have near zero in-out assortativity), we see that both the structure-based and hybrid selection processes result in disassortative networks before the phase transition, with the former continuing to show weak disassortativity after the transition, while the latter is nearly neutral after the transition. The random and structure-based processes result in no correlation between degree and sensitivity while, as expected, the sensitivity-based selection process creates a negative correlation between a node's out-degree and its sensitivity. The hybrid process shows weaker and non-monotonic negative correlations.	29
2.6	In-out assortativity for the transcriptional gene regulatory networks listed in Table 2.1, as a function of the number of links added. Links are ordered using gene age information [1], and a randomized ordering of the links in the final network is shown for comparison. Both orderings yield the same final network. <i>E. coli</i> is omitted, due to lack of gene age information from [1]. Note that the horizontal axis is scaled by total number of final links (E_{final}), rather than number of genes (N), to allow for comparison across the different networks.	33

3.1	Left: The critical boundary for the sync model (3.1) with Gaussian noise $\eta_t^k \sim \mathcal{N}(0, \sigma^2)$. Values of (κ, σ) above the blue curve lead to an incoherent system state at equilibrium ($\rho_\infty \approx 0$). Colored dots mark values of (κ, σ) used in the example steady-state distributions shown in the right figure, with colors of the dots matching colors of the distributions. Right: Example steady-state distributions for the sync model (3.1) with Gaussian noise $\eta_t^k \sim \mathcal{N}(0, \sigma^2)$ for different values of σ and fixed κ , generated from a simulation of $N = 10^5$ oscillators. Note that each distribution has been shifted to have mean 0. As σ decreases, the steady-state distribution changes from a uniform distribution (the incoherent state) to a symmetric, unimodal distribution (the coherent state), with the width of the distribution decreasing as σ continues to shrink.	40
3.2	Transition diagram for the contagion model.	43
3.3	Example outcomes of the contagion model for different choices of transmission coefficient β and recovery rate γ . In general, larger values of β lead to greater exponential growth of the infected population (and decay of the susceptible population), while larger values of γ lead to faster exponential decay of the infected population (and growth of the removed population). In the case where $\beta/\gamma < 1$ (top left), the infection immediately dies out.	44
3.4	Transition diagram for the sync-contagion model. We have updated the transition diagram from Figure 3.2 with the effect of the acceptance function $\alpha(\theta)$	46
3.5	Plot of the acceptance function $\alpha(\theta)$ used in our simulations. Individuals with orientations near the target orientation $\theta_\odot$ are more likely to take on the infection when exposed.	46
3.6	Example behavior of an incoherent (left) and coherent (right) system. Simulations used a population of $N = 10^4$ oscillators. Top row: Fractions of the population in each category (susceptible, infected, and removed) over time. Middle row: Distribution of orientations over time. The red center line indicates $\theta_\odot = 0$. The mean angle ϕ_t is marked by a yellow dot. Bottom row: The coherence of the full population ρ_t , the infected subpopulation $\rho_t^{\mathcal{I}}$, and the uninfected subpopulation ρ_t^{SUR}	54
3.7	Distributions of orientations over time for coherent systems with different choices of noise distribution width σ and $\kappa = 0.5$, $\beta = 0.1$, and $\gamma = 0.01$. The mean angle ϕ_t is marked by yellow dots, and the central red line indicates $\theta_\odot = 0$. As the width of the noise distribution increases, the uninfected population realigns towards $\theta_\odot$ at a faster rate, at the cost of a less tight distribution.	56

3.8	Heatmaps of our measures in $\kappa\beta$ -space for low noise ($\sigma = \pi/18$, left) and moderate noise ($\sigma = \pi/9$, right). Results are averaged over five realizations each with a system of $N = 1000$ oscillators. The vertical dashed red line indicates the critical value of κ_c for each value of σ . Incoherent systems exist to the left of the red line, while coherent systems exist to the right. Top row: Absolute value of the final mean angle ϕ_∞ scaled by π . For incoherent systems, the final mean angle is a random value, leading to a mean near 0.5. Middle row: The maximum size of the infected population $I_{\max}$ as a fraction of the total population. Bottom row: The maximum mean cosine $C_{\max}$, which takes into account both the mean orientation ϕ_t and the coherence ρ_t of the distribution.	58
3.9	Our measures for fixed values of $\beta = 0.3$, $\sigma = \pi/18$, and $\gamma = 0.01$ as κ varies. The vertical dashed red line indicates the critical value of κ_c for $\sigma = \pi/18$. For coherent systems ($\kappa > \kappa_c$), intermediate values of β lead to a drop in performance according to all three measures. Within this region, the distribution is not wide enough (κ too high) and does not realign fast enough (κ too low) to be within an appropriate region for the infection to continue, as required by the acceptance function $\alpha(\theta)$	59
3.10	Our three measures for different varying values of β on incoherent systems with $\kappa = 0.02$, $\gamma = 0.01$, and $\sigma = \pi/9$. Results are simulated on a system of $N = 1000$ oscillators and averaged over 100 realizations. The large grey box marks the region where $\beta\alpha_o/\gamma < 1$, where the initial infection does not spread. The final mean orientation in an incoherent system is a random value, which leads to $ \phi_\infty /\pi$ averaging out to 0.5 for all values of β . The maximum size of the infected population and the maximum mean cosine measure both increase as the transmission coefficient increases.	60
3.11	Distributions of orientation over time for $\beta = 0.2$ (left) and $\beta = 0.9$ (right) for two coherent systems with $\kappa = 0.4$, $\sigma = \pi/18$ and $\gamma = 0.01$. The mean orientation ϕ_t is indicated by yellow dots, and the horizontal red line indicates $\theta_\odot = 0$. When β is made too large, the exposures occur too quickly, leading to most exposures turning into instant removals, due to $\alpha(\theta)$. The transmission coefficient must be slow enough to allow for partial realignment of the uninfected population before the majority of exposures occur, while still large enough to sustain the infection at low levels.	61
3.12	Classifications of sync-contagion model behaviors, experimentally derived from the results shown in Figure 3.8. The classification regions are as follows: infectionless (black), temporary realignment (red), full permanent realignment (green), partial permanent realignment (yellow), and limited/no realignment (grey). Partial permanent realignment is based on ϕ_∞ being within $\pi/6$ of the target orientation. Similarly, limited/no realignment is based on ϕ_∞ being further than $5\pi/6$ from the target orientation.	62

4.1	Top: New COVID-19 cases $x(t)$ for the United States t days after March 1, 2020 shown from March 21, 2020 ($t = 20$) to February 28, 2021 ($t = 364$). Bottom: Transformations of the U.S. case data using the windowed average (blue) and the local linear fit (orange) methods for different choices of m . For $m = 1$, the two transformations are identical, but even for larger m the transformations give similar results. Note that larger m leads to greater ‘smoothing’ of the data, causing increased suppression of the oscillations.	72
4.2	Top: New COVID-19 cases $x(t)$ for New York (left) and Kansas (right) t days after March 1, 2020 shown from March 21, 2020 ($t = 20$) to February 28, 2021 ($t = 364$). Bottom: Transformations of the New York and Kansas case data using the windowed average (blue) and the local linear fit (orange) methods for different choices of m . Note that the two transformations are identical for $m = 1$. For larger values of m , the transformations give similar results for well-behaved data, like New York and the United States data (Figure 4.1). For poor-quality data, such as for Kansas, the two transformations are more distinct. The windowed average approach can more effectively remove massive spikes, but is less effective at overall suppression of the resulting oscillations (see for example $m = 6$).	73
4.3	The structure of a standard reservoir computer. During ‘open loop’ training, the switch on the left takes in the input $\mathbf{u}(t)$. The input is coupled to the reservoir through the input matrix $\mathbf{W}_{\text{in}}$ and update Eq. (4.9). The output is then generated by multiplying the reservoir state $\mathbf{r}(t + 1)$ by the output matrix $\mathbf{W}_{\text{out}}$. After training, the switch on the left is rotated to accept the output of the reservoir computer as the input for the next time step, creating an autonomous ‘closed loop’.	75
4.4	The two stages of the library-augmented reservoir computing process. In stage one (top), a library of trained reservoir computers are used to train an autoencoder to predict the pair $(\mathbf{W}_{\text{out}}, \mathbf{r}_0)$, consisting of an output mapping $\mathbf{W}_{\text{out}}$ generated from training the RC and an associated reservoir state $\mathbf{r}_0$ at the end of training. In stage two (bottom), a state from the embedded space of the autoencoder is decoded to generate a pair $(\tilde{\mathbf{W}}_{\text{out}}, \tilde{\mathbf{r}}_0)$. This pair is used to construct a new reservoir computer, that is otherwise identical to those in the library. The sample reservoir computer generates a signal that is compared to a new (short) signal that we wish to predict. Using an optimization scheme, information is sent back to the decoder to pick a new pair $(\tilde{\mathbf{W}}_{\text{out}}, \tilde{\mathbf{r}}_0)$ that better matches the new data until some stopping criterion is reached.	79
4.5	Reservoir computer predictor results for different choices of forgetting factor λ . For each value of λ shown, we increment through all prediction start times t_0 in $[50, 150]$. For each value of t_0 , a reservoir computer was trained on the entire history (i.e., all data from $t = 0$ to $t = t_0 - 1$), with data transformed according to the percent daily change Eq. (4.7).	82

4.6	The true daily U.S. COVID-19 case data (black) and example predictions for the four prediction methods: reservoir computer with the windowed average transformation applied to daily cases with $m = 6$ (magenta), reservoir computer with the local linear fit transformation applied to daily cases with $m = 6$ (cyan), a linear fit applied to the daily case data using 10 points (orange), and a quadratic fit applied to the daily case data using 12 points. Predictions start at times $t = 70$ (top) and $t = 110$ (bottom). Vertical dashed lines mark the start and end of data points ‘visible’ to the corresponding predictors.	84
4.7	State-level COVID-19 case data (black) compared to a reservoir computer prediction using the local linear fit transformation (cyan) with $m = 6$ and a linear extrapolation (orange) using 10 points. All predictions start on day 70, the first day after the vertical dashed black line. The vertical orange dashed line marks the start of points used by the linear extrapolation and the vertical cyan dashed line marks the start of points used to train the reservoir computer. Here we show the first twenty-six states alphabetically and the District of Columbia. The remaining states are shown in Figure 4.8.	85
4.8	The continuation of Figure 4.7, showing state-level COVID-19 case data (black) compared to a reservoir computer prediction using the local linear fit transformation (cyan) with $m = 6$ and a linear extrapolation (orange) using 10 points. All predictions start on day 70, the first day after the vertical dashed black line. The vertical orange dashed line marks the start of points used by the linear extrapolation and the vertical cyan dashed line marks the start of points used to train the reservoir computer.	86
4.9	Predictor statistics for the four different predictor methods with different choices of m for the reservoir computer transformations (a,d) and number of points for the polynomial fits (b,e) applied to the U.S. COVID-19 daily case data. Statistics are taken by generating a prediction for each starting time t_0 in $[50, 250]$. In (c,f), we compare the distributions for the errors of the reservoir computer predictions using the local linear fit transformation with $m = 6$ and the linear extrapolation using 10 points.	88
4.10	Predictor statistics for the four different predictor methods with different choices of m for the reservoir computer transformations (a,d) and number of points for the polynomial fits (b,e) applied to the New York COVID-19 daily case data. Statistics are taken by generating a prediction for each starting time t_0 in $[50, 250]$. In (c,f), we compare the distributions for the errors of the reservoir computer predictions using the local linear fit transformation with $m = 6$ and the linear extrapolation using 10 points.	89

4.11	Predictor statistics for the four different predictor methods with different choices of m for the reservoir computer transformations (a,d) and number of points for the polynomial fits (b,e) applied to the Kansas COVID-19 daily case data. Statistics are taken by generating a prediction for each starting time t_0 in $[50, 250]$. In (c,f), we compare the distributions for the errors of the reservoir computer predictions using the local linear fit transformation with $m = 6$ and the linear extrapolation using 10 points. Note that the box plots for the reservoir computer predictors (a,d) with low values of m extend way beyond the vertical scale used. A zoom out is provided in Figure 4.13.	90
4.12	Scatter plots comparing the one-week cumulative error (left) and one-week NMAE (right) for the reservoir computer prediction using the local linear fit transformation with $m = 6$ and the linear extrapolation using 10 points. The coordinates of each point are the median values of the error metrics for the labeled locations. The red line marks $y = x$. The plot includes forty-seven states, the District of Columbia, and the United States as a whole. Not included are Connecticut, Rhode Island, and Vermont, which each had significantly higher median errors from both predictors.	91
4.13	Zoom out of Figure 4.11(a,d), showing the full distributions for low values of m	92
4.14	Embedding of the U.S. COVID-19 library into a three-dimensional latent space using an autoencoder. The library consists of reservoir computers trained over 50-day overlapping windows of data from $t = 30$ to $t = 300$. Daily case data were transformed using the percent daily change Eq. (4.7) before input into the RCs. Points are colored according to the first day of the training window, with green points corresponding to starting in $[30, 100]$, blue points corresponding to starting in $(100, 200]$, and red points corresponding to starting in $(200, 300]$. Panels (a–c) are projections of the points onto the three coordinate planes. Panel (d) shows all points in three-dimensions.	94
4.15	Predictions using the library-augmented reservoir computing method (red lines) using five points of training data compared to a single reservoir computer (blue lines) using 21 days of training data and a linear extrapolation (orange lines) using five days of data. Vertical red dashed lines indicate the start of the LARC training data and data used for the linear extrapolation, and vertical blue dashed lines mark the start of the single RC training data. Predictions start on the first day after the green dots. Each bottom panel is a one-week zoom in of the predictions shown in their corresponding top panels. The library used for LARC consists of RCs trained on 50-day overlapping windows in $[30, 125]$	96

4.16	Embedding of the New York COVID-19 library into a twelve-dimensional latent space using an autoencoder. The library consists of reservoir computers trained over 50-day overlapping windows of data from $t = 30$ to $t = 300$. Daily case data were transformed using the percent daily change Eq. (4.7) before input into the RCs. Each panel is a projection onto a plane spanned by a pair of coordinate axes. Points are colored according to the first day of the training window, with green points corresponding to starting in $[30, 100]$, blue points corresponding to starting in $(100, 200]$, and red points corresponding to starting in $(200, 300]$	98
4.17	Embedding of the combined U.S. and New York COVID-19 library into a twelve-dimensional latent space using an autoencoder. The library consists of reservoir computers trained over 50-day overlapping windows of data from $t = 30$ to $t = 300$. Daily case data were transformed using the percent daily change Eq. (4.7) before input into the RCs. Each panel is a projection onto a plane spanned by a pair of coordinate axes. Top: Points are colored according to the first day of the training window, with green points corresponding to starting in $[30, 100]$, blue points corresponding to starting in $(100, 200]$, and red points corresponding to starting in $(200, 300]$. Bottom: Points are colored according to region, with red indicating New York and black indicating the U.S.	99
A.1	The fractions of nodes connected by a directed path of length two connected by at least one other directed path of length two.	103
A.2	Growth of the connected components on the full network (left) and the damaged subnetwork (right). Results shown are the averages of 100 runs on networks of size $N = 1000$. We omit GIN and GIN* due to their symmetry with GOUT and GOUT*, respectively.	104

List of Abbreviations

BT	bow tie
GBT	giant bow tie
GIN	giant in-component
GOUT	giant out-component
GSCC	giant strongly connected component
IN	in-component
LARC	library-augmented reservoir computing
OUT	out-component
NMAE	normalized mean absolute error
RC	reservoir computer
RLS	recursive least squares
SCC	strongly connected component
SIR	susceptible-infected-recovered

Chapter 1: Introduction

Network models have proven to be useful tools in representing the complex interactions of many-agent systems, especially when combined with a dynamical system model to describe the state of the system over time. These systems are made further complex when we no longer assume that the ‘rules’ of the dynamics are fixed, but instead changing over time. Such systems quickly become intractable to study analytically, but can instead be studied through computer-simulated experiments to uncover their unknown behaviors.

In this thesis, we focus on analyzing three dynamical networks that apply to biological or social systems (or both, in the case of epidemic models). Each dynamical network features rules that are ever-changing (adapting) in response to other properties of their systems. We quickly summarize each investigation below, with more detail present in the remainder of this chapter:

1. In our first investigation, we consider a model of gene network evolution in which the topology of the network changes over evolutionary time as new links are added via a selection rule that uses some combination of structural and dynamical information of the network. The changing topology inherently affects the possible dynamics of the network, eventually leading to widespread network failure. We are interested in the location of the critical point at which failure occurs and which

selection rules perform best at delaying this critical point.

2. In our second investigation, we combine a phase synchronization model with a contagion model using a bidirectional feedback. In other words, the behavior of the synchronization model depends on an individual's infection status, and the spread of the infection depends on an individual's phase. We work to classify all possible behaviors of this system, considering both temporary and permanent consequences.
3. In our third investigation, we use a data-driven approach through reservoir computing to forecast COVID-19 cases in the United States. The disease dynamics depend on the constant change in human behavior, both naturally occurring and following government-mandated orders, in response to reported case numbers. In addition to generating case predictions with a reservoir computer, we observe the result of embedding several trained reservoir computers in a latent variable space through an autoencoder. We find that reservoir computers trained on similar time series form clusters in the latent space, indicating the possibility of a low-dimensional underlying system.

The work from investigation one has been published in J. R. Soc. Interface [2]. The work from investigation two will be submitted for publication to J. R. Soc. Interface with Brandon Alexander (author of this thesis) as the lead author and Michelle Girvan (thesis advisor) as the senior author. The work from investigation three was done in collaboration with Andrew Pomerance, Daniel Canaday, Brian Hunt, Edward Ott, and Michelle Girvan. Brandon Alexander took the lead on all work presented here. Daniel Canaday led the effort in developing the library-augmented reservoir computing approach, which will be

published in a separate paper.

1.1 Phase transitions and assortativity in models of gene regulatory networks evolved under different selection processes

Gene regulatory networks are believed to exist near a critical point, where they have maximized dynamical versatility without risk of widespread network failure. While gene networks have been extensively studied as they appear today, not much is known about how they evolved to this point and what features may have guided this evolution. In Chapter 2, we propose a model of gene network evolution, based on a Boolean network model, that grows over evolutionary time based on the topological and dynamical properties of the network.

In a Boolean network model for gene regulation [3], each gene is represented by a node with regulatory interactions between pairs of genes represented by directed edges. A node is either in the state ‘1’ (the gene is expressed) or ‘0’ (the gene is not expressed). At each time step, all nodes are simultaneously updated according to the states of their inputs and a set of rules in the form of a truth table. For a finite-sized network with deterministic update rules, the dynamics will always reach a steady-state of a fixed point or a periodic orbit. The stability of the dynamics can be quantified by considering how much the steady-states differ when beginning with two initially-close network states (i.e., a single node is switched between 0 and 1).

Determining the stability of a Boolean network through direct simulation is a computationally intensive task. However, if we consider a locally tree-like network under the

‘semi-annealed’ approximation [4, 5], then the stability problem can be directly mapped onto a percolation problem, which is considerably faster to simulate. Specifically, if we consider the sensitivity of a node, defined as the probability of the node changing state if at least one of its input changes, and set that as the probability that the node is ‘filled-in’, then the stability measure for the network as a whole is given by the size of the giant out-component of the filled-in subnetwork.

For our model of gene network evolution, we begin with a network of nodes with no edges. Each node is assigned a sensitivity and filled-in accordingly. For each generation of network evolution, we select an edge to add to the network according to a competitive selection rule, motivated by natural selection and modeled after the rules introduced by Achlioptas *et al.* [6]. After an edge has been added, we track the size of the giant out-component of the filled-in subnetwork, which is equal to the stability measure for the entire network. This process is repeated for several generations, until we sufficiently pass the critical point at which the network switches from stable to unstable dynamics (the percolation transition of the filled-in subnetwork).

Our main focus is determining what selection rules are more effective at delaying the onset of percolation, and thereby delaying instability. We consider four types of rules: random selection, structure-based selection, sensitivity-based selection, and hybrid (structure and sensitivity) selection. We find that topological information, i.e., adding connections that keep components small, is more effective at delaying percolation than avoiding connection between highly-sensitive (likely to be filled-in) nodes.

For further validation, we analyze gene network data from five organisms. For each network, we approximate the order of edge addition by ordering the genes by their

estimated ages. By tracking the in-out assortativity of the networks over evolutionary time, we observe a striking similarity to the in-out assortativity of our simulated networks grown under structure-based and hybrid rules.

1.2 The sync-contagion model of dual information spread

Synchronization is a natural phenomenon found in social systems of both humans and animals. In many cases, the ‘status quo’ synchronization of a group will adapt to the presence of some external stimulus. Consider, for instance, the swarming behavior of a group when presented with a threat. Whatever the default behavior may be, initial ‘leaders’ aware of the threat will alert others, who will follow the leaders to escape. In such scenarios, two systems are at play: a synchronization system that determines how strongly individuals match each others’ behaviors and a contagion system that controls the spread of the alert through the population. Furthermore, the two systems are coupled, where synchronization behavior changes based on one’s knowledge of the alert. In Chapter 3, we present a coupled sync-contagion model that describes the interplay of these two systems. Other than group escape in response to a threat, we also describe an application to opinion dynamics, in which individuals’ opinions adapt in response to a hot topic driven by a group of promoters.

We base our model on that of Chicoli and Paley [7], originally used to represent fish school response to the presence of a predator. Among several minor adjustments, we make two key changes to their model. First, the Chicoli and Paley model only contains unidirectional feedback between the sync and contagion dynamics. Specifically,

the contagion status of an individual (susceptible, infected, or removed) determines the synchronization behavior, but not the reverse. We keep this interaction but also add an ‘infection acceptance function’ $\alpha(\theta)$ that creates a *bidirectional* feedback, by allowing the sync dynamics to influence the contagion spread. The acceptance function gives the probability of a susceptible individual to become infected upon being exposed, depending on their orientation θ , with orientations further from some target orientation having lower acceptance probabilities. In the case of a failed infection upon exposure, the individual immediately becomes removed (not infected and can never become infected). In the context of swarming, the acceptance function takes into account individuals who are ‘facing the wrong direction’ and are unable to either see or turn quickly enough to join in instantaneous realignment. For our opinion dynamics application, the acceptance function filters out individuals whose opinions differ too greatly from the hot topic and choose not to become promoters. The acceptance function introduces a ‘barrier’ to realignment and drives several of the emergent behaviors we observe.

For the second key change, we note that the Chicoli and Paley model distinguishes between naturally coherent systems (attentive obligate shoaling) and naturally incoherent systems (attentive facultative shoaling) through setting the initial orientations of the population to either be polarized or uniformly distributed, respectively. In both cases, the same coupling strength κ is used for the sync dynamics that will result in synchronization of the systems over time. We choose to instead distinguish between coherent and incoherent systems by varying this coupling strength κ . While incoherent systems will no longer be able to permanently realign, we can determine the ability of these systems to temporarily realign while also actively dispersing.

We find that the overall behavior of our sync-contagion system can be classified into five categories. The first category consists of the trivial case where no initial contagion spread occurs, due to the exposure rate being too slow. The second category consists of all naturally incoherent systems, which are able to temporarily realign towards the target orientation, but will eventually return to a uniform distribution. The final three categories are various levels of permanent realignment success in coherent systems: no/limited realignment, partial realignment, and full realignment. Which of these last three categories a system falls into depends primarily on the relative strengths of the contagion exposure rate and the sync coupling.

1.3 A reservoir computing approach to forecasting COVID-19 data

The COVID-19 pandemic presented the world with a global health crisis never before seen in our lifetime. In response, countless efforts have gone towards forecasting COVID-19 cases to better plan emergency response. While the most successful predictions have been model-based, considerable effort has gone into data-based model-free approaches using machine learning. In Chapter 4, we use a reservoir computing approach to forecast United States COVID-19 cases at both the national and state levels.

A reservoir computer is a system in which an input signal is mapped into a high dimensional recurrent neural network (the reservoir), which is then trained to predict some output signal. When applied to time series forecasting, the output signal is the input at the next time step. Reservoir computers offer a level of simplicity in their training. The input mapping and reservoir topology are completely randomized, with the output mapping

being the only piece affected from the training phase. In training, data are fed into the reservoir with the resulting sequence of reservoir states tracked. The mapping from the reservoir to the desired output is then determined through a least squares fit or similar method. Once trained, the output of the reservoir computer can be used as its input at each subsequent step, creating the ‘closed loop’ fully autonomous prediction mode. Reservoir computers have proven to be powerful tools in forecasting and mimicking properties of select chaotic systems.

Most applications of reservoir computers have focused on forecasting model systems, in which training data is generated from a system of differential equations and can be produced with arbitrary size and resolution. By contrast, real-world data is rarely as available or high-quality. Particularly, with daily case reports, a full year of COVID-19 case data is substantially smaller than usually required for successful machine learning training. The case data are also heavily influenced by testing and reporting frequency, leading to a weekly oscillation in case numbers, or in some regions, several days of apparently zero cases interrupted by artificial ‘spikes’ in cases.

To address the reporting irregularity, we find it necessary to apply a preprocessing transformation that will ‘smooth’ out the data. The contributions from a single day are then spread out over multiple days. This transformation is especially needed in states that report cases once every few days, in order to remove the massive spikes that would otherwise be impossible to predict. In addition, the data must be detrended to remove the non-stationary behavior of the mean. We must choose an appropriate preprocessing transformation that will also convert the data to be approximately constant.

We compare the short-term predictions from our reservoir computer to those from

a linear and quadratic fit. The quadratic fit performs the worst of these methods, due to its tendency to deviate (quadratically) from the true data whenever the fit is poor. We also find that the reservoir computer is able to occasionally replicate the oscillations present in real case data, but when considered as a whole, is unable to significantly outperform a linear regression.

While our preprocessing steps address the data quality, we also need to resolve the lack in data quantity. As one potential solution, we consider the library-augmented reservoir computing (LARC) approach. In the LARC approach, we construct a ‘library’ of reservoir computers trained on COVID-19 data. The library is then embedded in a low-dimension latent space by an autoencoder. When presented with new training data, rather than directly training a new reservoir computer, a point in the latent space is selected and decoded to produce the reservoir computer. Through an optimization scheme, selection of the point in latent space can be refined until a best match is produced.

Success of LARC requires reservoir computers produced from similar time series to be located close in the latent space. We demonstrate that such clustering is present both temporally, where RCs trained on time series segments from similar start times are grouped together, and regionally, where time series from the same geographical zone are grouped together.

Chapter 2: Phase transitions and assortativity in models of gene regulatory networks evolved under different selection processes

This chapter has been published in the April 2021 issue of the Journal of the Royal Society Interface [2]: Brandon Alexander, Alexandra Pushkar, and Michelle Girvan. Phase transitions and assortativity in models of gene regulatory networks evolved under different selection processes. J. R. Soc. Interface., 18(177):20200790, 2021. (doi: 10.1098/rsif.2020.0790).

2.1 Introduction

Mathematical models of gene regulatory networks provide a powerful tool for understanding the complex features of genetic control. While various modeling efforts have been successful at explaining gene expression patterns, much less is known about how evolution shapes the structure of these networks. Recent studies suggest that evolutionary ‘tinkering’ plays a large role in the organization of biological networks [8]. Further, these networks are often thought to exist near some critical point [9–11], where dynamic variability is maximized without reaching widespread network failure/breakdown. The phase transition between stability and instability in networks has been widely investigated, with studies focusing on the effects of topological features [3, 12–14], dynamical

features [4, 15–17], or both [18] and their contributions to the location and behavior of the transition. Of particular interest is the evolutionary process that leads to this critical point and how this process depends on both the topological and dynamical properties of the network and its nodes. Some previous studies have considered the evolution of these networks but begin with already-established networks that are then allowed to change over time [19–21]. By contrast, we consider the process from the very beginning, starting with an ‘empty’ network.

In order to gain insight into this process, we study simple models of gene network evolution in which links are added according to various competitive selection rules. Similar rules were explored by Achlioptas *et al.* [6], who demonstrated that undirected networks grown following competitive link selection can have drastically delayed percolation transitions, leading to a seemingly discontinuous ‘explosive’ transition, a topic which has since become popular in the network science literature [14, 22–27]. Squires *et al.* extended the idea of competitive selection to directed networks, where similar, but less drastic ‘weakly explosive’ transitions occurred [14]. Incrementally growing networks according to competitive selection rules is a natural choice for modeling biological networks, which are the result of generations of improvements through the complex process of natural selection. For the same reason, biological networks share commonalities with systems that are said to exhibit highly optimized tolerance, in which a yield or fitness is to be maximized [28, 29]. In such systems, competitive selection rules can be used heuristically to maximize a yield dependent on the size of a giant component. Besides evolution through natural selection, biological systems develop over time through neutral evolution, or evolution through mutation without selection [30–33]. While we primarily focus on

networks evolved via selection, we will also consider a simple case of neutral evolution for comparison. Other approaches have used a combination of mutation and selection to evolve networks that maximize the topological entropy [34] or minimize the distance to a time-dependent network output function [20]. By comparison, our modeling efforts focus on growth starting from a blank state, serving to highlight how simple differences in selection rules can give rise to very different network structures and change the nature of phase transitions in the network.

Our models of network growth build upon various studies of Boolean models of gene regulation. From a dynamical perspective, Boolean networks have been widely used as mathematical models of gene regulation since their introduction by Kauffman in 1969 [3]. In these models, at any given time, nodes (genes) exist in one of two states: ‘on’ (1), meaning that the gene is expressed, or ‘off’ (0), meaning that the gene is unexpressed. In Kauffman’s original model, gene states are synchronously updated at each time step according to a truth table, in which the state of a gene at time t is determined by the states of all of its input genes at time $t - 1$. (While we connect our evolutionary modeling approach to the case of synchronous update, we also discuss how the approach may be connected to cases of asynchronous update.) By tuning system parameters, the models can exhibit a transition from stable dynamics to ‘chaotic’ (unstable) dynamics [3, 4, 12]. Stability here is measured by the saturated average Hamming distance, or fraction of genes that differ in state after a long time, between the trajectories of two close initial conditions. Since a finite network of size N has at most 2^N possible states and the dynamics are deterministic, the trajectories will eventually reach an asymptotic state in the form of a fixed point or periodic orbit. If the state of a gene in the asymptotic state differs between these two

trajectories at time t , the gene is said to be ‘damaged’ at time t [5].

As demonstrated by Squires *et al.*, we can map the stability of Boolean networks directly to a percolation problem under certain assumptions [5]. In a percolation problem, the size of a ‘giant component’ is tracked as a certain tuning parameter is varied, typically passing through a critical value where the giant component transitions from being vanishingly small to being a non-trivial fraction of the size of the network. Following Squires *et al.*, by setting a gene’s sensitivity, i.e., the probability that the gene changes state if one of its inputs changes state, equal to its probability of being ‘damaged’, the saturated average Hamming distance, which provides a measure of instability in the gene network dynamics, can be mapped directly onto the size of the largest out-component in the sub-network of ‘damaged’ nodes. As either the node sensitivities or network connections are tuned, the phase transition from dynamic stability to instability then corresponds to the percolation transition of this giant component.

We extend this idea by allowing the topology of our network to change over time following different competitive selection rules, which works as a primitive model of biological networks evolving via natural selection. As new links are added, at some critical point, percolation will occur in the damaged subnetwork, corresponding to the network’s transition from stability to instability.

To summarize, we are motivated by a model of gene regulatory dynamics in which there are benefits for the system to exist on the stable side of the transition between stability and instability. We utilize the mapping between a model of this kind and an appropriately defined percolation problem to explore simple models of evolution via growth. We note that the percolation problem can be considered as its own simple abstraction of

the propagation of failure in gene regulatory networks. Using the percolation framework, we investigate the following growth processes: (1) random link addition, as a control; (2) sensitivity-based competitive selection; (3) structure-based competitive selection; and (4) hybrid competitive selection. We find that incorporating network structure into the selection rule is key to delaying percolation of the damaged subnetwork, even while having no knowledge of each gene’s sensitivity, or probability of becoming damaged. In fact, incorporating information about sensitivity can backfire, leading to highly exclusive networks that would rather form large components than connect to high risk genes. Finally, we compare the results of our simulations to data for several empirically-derived gene regulatory networks and find similarities in structural properties.

2.2 Methods

2.2.1 Modeling failure propagation in gene networks as a percolation problem

We construct a model of gene regulation as follows: we first consider damage propagation on a network with a fixed topology, which will later be allowed to evolve. We are motivated by the traditional Boolean models of gene regulation in which each gene in the network has a fixed ‘truth table’, or output rule, that determines its state (0 or 1) at time t based on the states of its inputs at time $t - 1$. To simplify the problem for our purposes but still capture the effects of network topology, we apply the semi-annealed approximation [4], in which the network remains fixed, but the truth table for gene i is no longer fixed but rather randomly filled in at each time step according to some bias b_i ,

the probability that its state is 1. We can think of b_i as the expression bias of gene i , with genes having a wide distribution of expression biases. Working within the semi-annealed approximation, from the gene’s bias, we can then calculate its sensitivity $q_i = 2b_i(1 - b_i)$, which represents the probability that a gene changes state if the state of one or more of its inputs change. This can also be thought of as the probability that a gene spreads ‘damage’. Damaged genes are those whose states differ in the asymptotic states of two initially close trajectories of the network’s dynamics, meaning a small perturbation in the state of the network can lead to unpredictable dynamics of such genes, likely causing improper regulatory behavior.

Following [5], for a locally treelike network under the semi-annealed approximation, dynamic instability of the network, measured in terms of the average saturated Hamming distance between trajectories of two close initial conditions, is equivalently measured by the expected size of the giant out-component of the damaged subnetwork, which takes into account both the network topology and the genes’ propensities for spreading damage, i.e., their sensitivities. In particular, the phase transition from stability to instability occurs simultaneously as the onset of a non-trivial out-component of the damaged subnetwork (i.e., a component whose size does not go to zero as the number of nodes gets larger). In Appendix A, we show that our simulated networks display the relevant locally treelike properties.

While we are motivated by a model of synchronously updated genes with locally treelike connectivity, we note, importantly, that the semi-annealed approximation works well even for an asynchronous update scheme and for the case of networks with many small loops. Although there are important differences between synchronous and asyn-

chronous update models (e.g. the occurrence of limit cycles [35]), Refs. [4] and [18] demonstrate that the semi-annealed approach for determining the transition between global stability and instability (and the growth of perturbations) also applies for asynchronous update. Further, Ref. [4] shows that the approximation holds even for networks that deviate substantially from the locally treelike assumption. This is consistent with studies that show that calculations involving the locally treelike approximation provide accurate results even for networks that are very far from treelike [36, 37].

By using the semi-annealed approximation, we are able to work entirely in the realm of the percolation problem, without needing to directly simulate the complex dynamics of the Boolean network. In fact, we can think of the percolation problem as its own simple abstraction of gene regulatory networks, in which genes become damaged with different propensities and they can propagate that damage to their target genes.

To incorporate evolutionary changes into our percolation framework, we add new links (edges) individually to the network according to a given selection rule, described in Sec. 2.2.3. Each new link represents a new regulatory interaction that has evolved in the network, potentially changing the stability of the dynamics. We then repeat this procedure until a sufficient number of links have been added to the network.

For our simulations, we initialize the network to consist of N genes and no links. We let E denote the number of links (edges) in the network at later steps. The gene biases are drawn from a uniform distribution $b_i \sim U[0, 1]$. Note that for a single simulation, bias values do not change, meaning we are fixing an inherent dynamical property of each gene. This is a restrictive assumption, but it provides a nice starting point for comparing networks evolved under different selection rules. Before the simulated evolution process

begins, each gene is assigned damaged status with probability q_i , leading to an average of one-third of all genes being damaged (due to how biases were chosen). By averaging over multiple simulations, we found that keeping the same damage status constant throughout all steps of a single simulation produces very similar results to re-randomizing the statuses at each individual step, so we have chosen to keep the damaged statuses constant. Finally, at each step, we track the size of the giant components in both the full network and damaged subnetwork, which also allows us to efficiently simulate each growth process using the algorithm suggested in [14].

2.2.2 Tracking network components

For a gene i , define the *in-component* $\text{IN}(i)$ to be the set of all genes that can reach i via a directed path in the regulatory network. Similarly, define the *out-component* $\text{OUT}(i)$ to be the set of all genes that i can reach via a directed path. The *strongly connected component* $\text{SCC}(i)$ is the intersection of these two sets and the *bow tie* $\text{BT}(i)$ their union.

For a directed network, we say the *giant strongly connected component* (GSCC) is the largest strongly connected component. We then define the *giant out-component* $\text{GOUT} = \text{OUT}(\text{GSCC})$, the *giant in-component* $\text{GIN} = \text{IN}(\text{GSCC})$, and the *giant bow tie* $\text{GBT} = \text{BT}(\text{GSCC})$. In systems where percolation occurs, all four giant components form simultaneously at the same critical point [38].

Within our networks, we will also have a subset of genes that are ‘damaged’. The damaged subnetwork is the directed network consisting of these genes and any links that exist between them. We use an asterisk (*) to distinguish the components of the damaged

subnetwork from those of the full network.

2.2.3 Network growth via selection

To decide which link to add at each step, we use a class of competitive selection rules based on those introduced by Achlioptas *et al.* [6] and extended to directed networks by Squires *et al.* [14]. The form of these rules is as follows:

1. Consider m potential ‘source’ genes $\{i_1, \dots, i_m\}$ and m potential ‘target’ genes $\{j_1, \dots, j_m\}$. All genes are sampled uniformly from the network.
2. Select the directed link $i \rightarrow j$ such that

$$i = \arg \min_{i \in \{i_1, \dots, i_m\}} f_s(i), \quad j = \arg \min_{j \in \{j_1, \dots, j_m\}} f_t(j). \quad (2.1)$$

3. If the link $i \rightarrow j$ already exists in the network or if $i = j$, then a different link is selected starting back at step 1.

The choice of $f_s(i)$ and $f_t(j)$ are chosen so as to delay the percolation transition of GOUT*. We consider three forms of competitive selection (sensitivity-based, structure-based, and hybrid) and the case of random link addition as a control. As in the original Achlioptas study, the selection rules we consider rely only on local information about the candidate links. In Appendix A, we study the extreme case of $m = N$ for structure-based selection and demonstrate that, as expected, having access to global information when adding links serves to delay the transition for a much, much longer period of time.

2.2.3.1 Random selection

In the case of $m = 1$, we select a random link uniformly from all possible links. This is equivalent to the directed Erdős-Rényi process [14, 39]. Random selection serves as a simple baseline to which we can compare the performance and behavior of other selection rules. In addition, one can consider random selection as a model for neutral evolution.

2.2.3.2 Sensitivity-based selection

To consider the effect of sensitivity, we define

$$f_s(i) = q_i, \quad f_t(j) = q_j. \quad (2.2)$$

Genes with higher sensitivity values are more likely to be damaged. The strategy used by sensitivity-based competition is then to prevent growth of the damaged subnetwork by preventing ‘risky’ genes from connecting.

2.2.3.3 Structure-based selection

To consider the effect of network structure in delaying the onset of percolation, we define

$$f_s(i) = |\text{IN}(i)|, \quad f_t(j) = |\text{OUT}(j)|. \quad (2.3)$$

This rule is a generalization of the dCDGM rule applied to directed networks [40]. The strategy imposed by structure-based competition is to prevent percolation in the damaged

subnetwork by preventing large components from forming in the full network.

2.2.3.4 Hybrid selection

The final rule is a combination of structure-based competition and sensitivity-based competition. We set

$$f_s(i) = q_i \times |\text{IN}(i)|, \quad f_t(j) = q_j \times |\text{OUT}(j)|. \quad (2.4)$$

The hybrid strategy will prevent components from becoming large in the full network, while also showing a preference towards connecting low risk (low sensitivity) genes. High sensitivity genes will only be accepted if they belong to small components.

2.3 Results

2.3.1 Visualizing networks evolved via different selection rules

For each of our network measures, we use the average node degree E/N as the tuning parameter. We can first gain intuition about the structure of our networks by considering smaller networks of size $N = 100$, as is done in Figs. 2.1 and 2.2. We begin by considering the networks when $E/N = 1.75$, a point after which percolation of GOUT occurs for both the random and sensitivity-based selection rules, but not the others. Compared to random selection, sensitivity-based competition has a preference towards connecting low sensitivity genes, even going so far as to exclude a large number of high sensitivity genes from the network, an effect that is more exaggerated for higher values

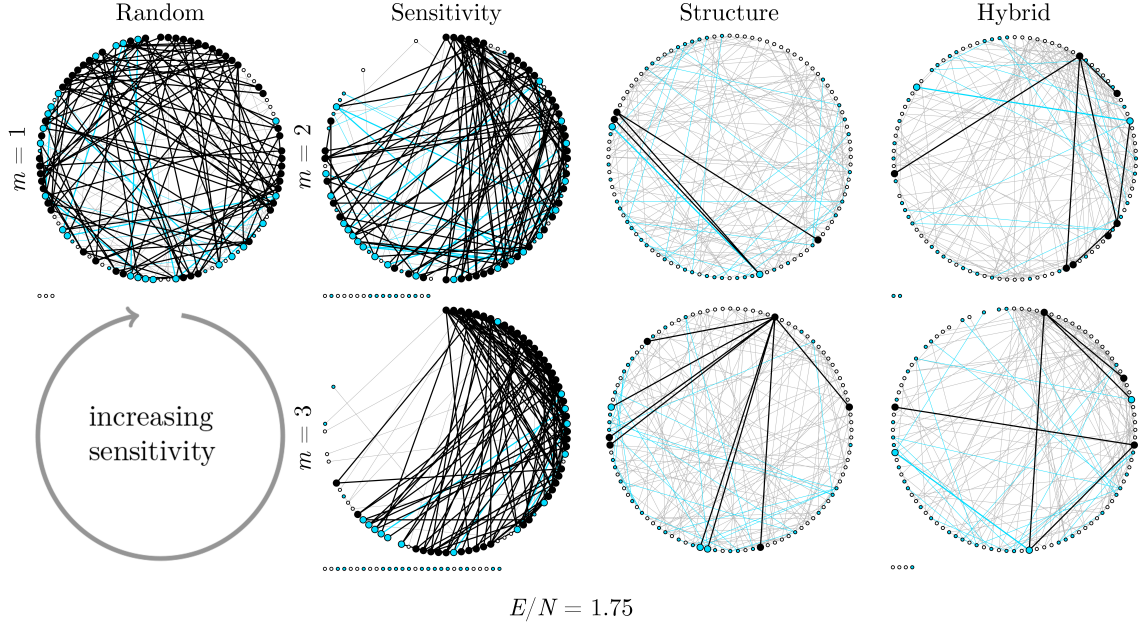


Figure 2.1: Structure of networks formed under our four different selection rules on a network of size $N = 100$, stopped after $E/N = 1.75$. Nodes are sorted according to sensitivity with least sensitive at the top of the circle and increasing in sensitivity clockwise. If a node is damaged, it is filled in blue. Nodes in GOUT are enlarged and filled in black (if undamaged) or blue (if damaged); all other nodes are filled in white. Isolated nodes with no links are placed below each network with their corresponding locations in the circular layout left empty. Thick links connect nodes in GOUT. Links are blue if they connect two damaged nodes, black if they connect nodes in GOUT that are not both damaged, or gray otherwise. We see that GOUT is much larger for the random and sensitivity-based processes as compared with the structure-based and hybrid processes. Both the sensitivity-based and hybrid processes show a higher density of links between low sensitivity nodes, which is much more pronounced in the former.

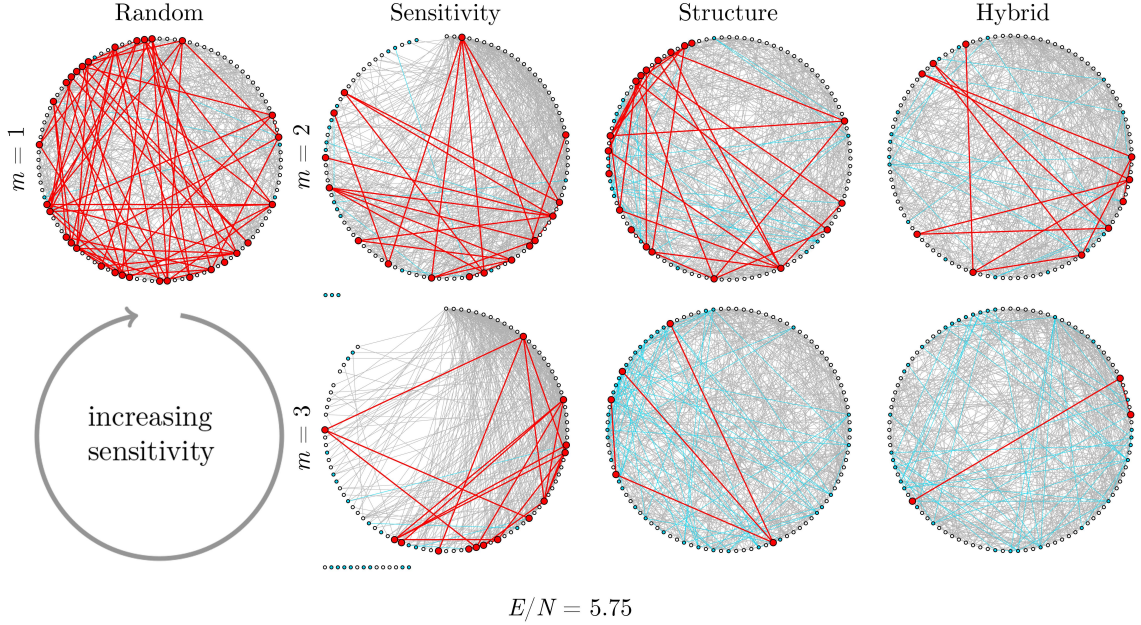


Figure 2.2: Network structure for a network of size $N = 100$ grown under the four selection rules stopped after $E/N = 5.75$. As in Fig. 2.1, nodes are sorted according to sensitivity, with least sensitive at the top of the circle and increasing in sensitivity clockwise and with isolated nodes left out of the circular layout and placed below each network. We now focus on GOUT* rather than GOUT. Nodes belonging to GOUT* are enlarged and red with thick red links connecting between them. Damaged nodes that do not belong to GOUT* are blue, with blue links connecting pairs of damaged nodes. All other nodes are colored white, and all other links are colored gray. Even though the size of GOUT for the sensitivity-based process is similar to the random process, the former has a much smaller GOUT*. The structure-based and hybrid processes show even smaller GOUT*, despite a large number of connections between damaged nodes.

of m . By comparison, both the structure-based and hybrid competition rules takes much longer to form a GOUT component. These two methods are nearly indistinguishable, except the hybrid method mimics some properties of sensitivity-based competition, namely having higher link density between low sensitivity genes and exclusion of high sensitivity genes.

We next consider a higher parameter value of $E/N = 5.75$, where GOUT* has now formed for random and sensitivity-based selection and is beginning to form for

structure-based and hybrid selection. Because sensitivity-based selection avoids connecting to high sensitivity genes overall, the GOUT* component mainly consists of lower sensitivity genes. Since most damaged genes have higher sensitivities, this also leads to fewer links overall between damaged genes and a smaller GOUT* component than in the case of random selection. We also observe that in random selection and sensitivity-based selection, nearly every link between damaged genes belongs to GOUT*. By comparison, structure-based and hybrid selection have a large number of links between damaged genes, but only few belong to GOUT*, since the individual out-components are kept small and avoid connecting to each other.

2.3.2 Growth of giant components

Now equipped with better intuition about these selection methods, we turn our attention to growth of the connected components in the $N \rightarrow \infty$ limit. We can approximately achieve this goal by simulating on a network of size $N = 10^5$, shown in Fig. 2.3.

As before, we begin by considering the growth of the full-network components, ignoring damage status. As expected, the formation of GOUT is delayed much further in structure-based and hybrid selection methods, due to their nature of selecting links to maintain smaller component sizes overall. Meanwhile, sensitivity-based selection instead tends towards earlier formation of the giant component, with higher values of m pushing this transition earlier. By increasing m in structure-based selection, we are more likely to find a smaller component, and thus are able to spread out links more efficiently to avoid forming larger components. By comparison, in sensitivity-based selection, increasing

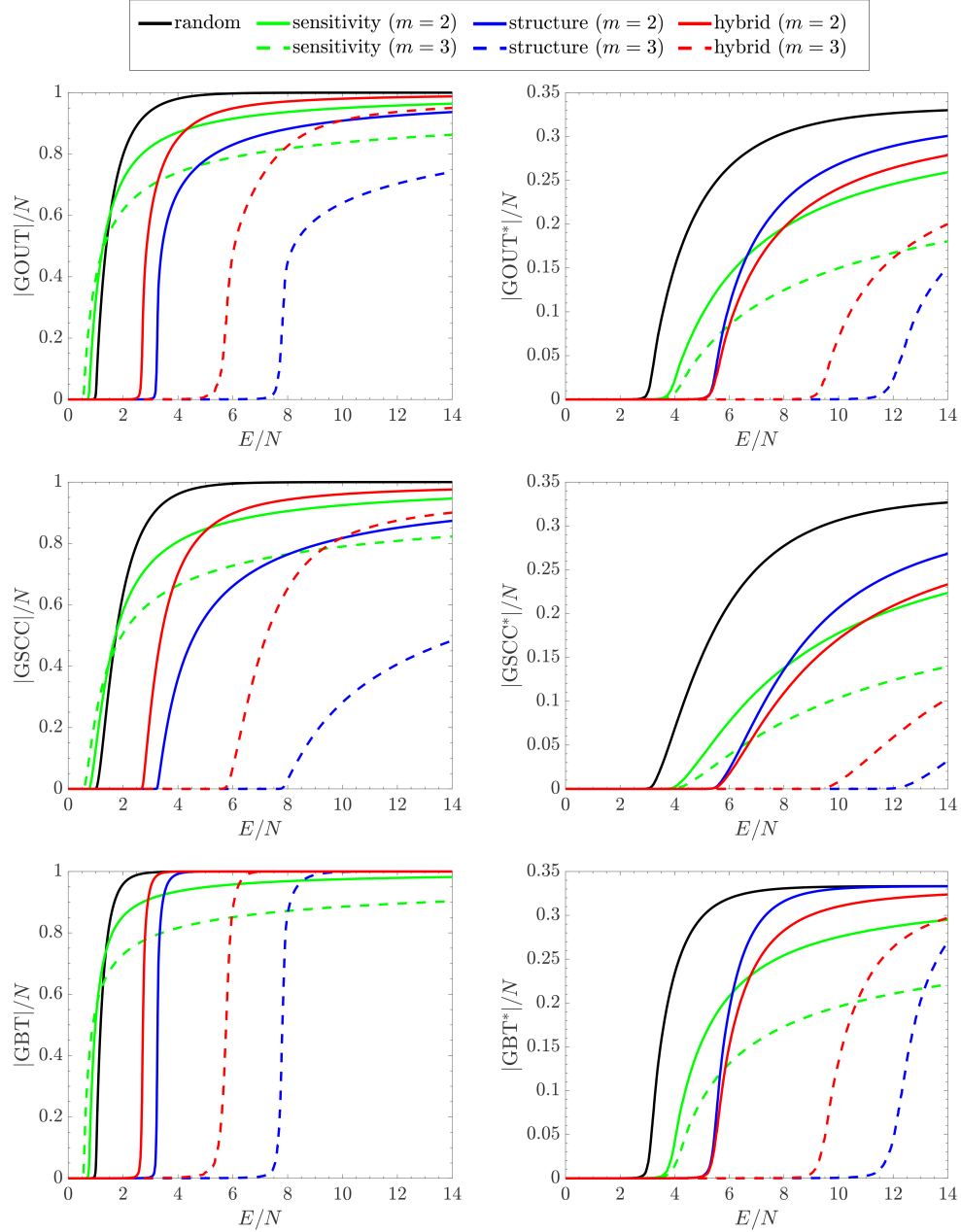


Figure 2.3: Growth of the connected components on the full network (left) and the damaged subnetwork (right). We compare growth under random selection to our three competitive rules. Results shown are the averages of 100 runs on networks of size $N = 10^5$. We omit GIN and GIN* due to their symmetry with GOUT and GOUT*, respectively. Of the selection rules considered, we see that structure-based selection (blue curves) does best at delaying the phase transition for the full network, while the structure-based and hybrid selection (red curves) exhibit simultaneous transitions when we consider only the damaged subnetwork, with the hybrid selection showing greater suppression of the component size after the transition.

m makes us more likely to find low sensitivity genes (and avoid high sensitivity genes altogether), resulting in a higher link density among the lower sensitivity genes and a faster formation of a giant component. The exclusion of highly-sensitive genes leading to faster emergence of a giant component also explains why hybrid selection transitions slightly earlier than structure-based selection.

Of greater interest is the formation of GOUT*. We again have structure-based and hybrid selection leading to later percolation of GOUT*, but unlike before, sensitivity-based selection leads to formation of GOUT* later than random selection. The sensitivity-based selection rule is designed to prevent formation of GOUT* by preventing connections between damaged genes, which leads to slower formation of GOUT* but not GOUT. Comparing structure-based and hybrid selection, we see that the critical point at which GOUT* forms is approximately the same when $m = 2$. But when m is increased, structure-based selection can delay the transition much more easily. Again, the additional effect introduced by considering sensitivities seems to be at play. Hybrid selection is willing to form larger components, as long as the genes involved have relatively low sensitivity. Eventually, however, the high sensitivity genes will have smaller components and be more favorable to gain a new link. It appears that this shift is enough to lead to faster growth of GOUT* than if all components were to be kept smaller, ignoring sensitivity.

Other than location of the percolation transition for GOUT*, we can also look at the growth rate of this component post-transition. One advantage of sensitivity-based and hybrid selection rules are their slow growth of GOUT*, despite the earlier transition. Since these rules will tend to prevent connections between damaged genes, a certain number of highly-sensitive genes will have trouble joining with GOUT*, and the rate at which

GOUT* can achieve its maximum size is reduced. In particular, increasing m will slow down the growth even further in sensitivity-based and hybrid selection, since larger values of m lead to increased exclusion of genes.

2.3.3 Maximizing yield

Our model exhibits certain features common to systems developed through *highly optimized tolerance* [28, 29], namely incremental improvements in response to maximizing some yield or fitness function, leading to specialized structures. While we do not directly maximize any yield function, our selection methods can be thought of heuristics for maximizing

$$Y = \frac{1}{N}(n_{\text{active}} - |\text{GOUT}^*|), \quad (2.5)$$

where n_{active} is the number of genes with at least one link. Only genes that interact with others will contribute to the dynamics in a meaningful way. Therefore, we should maximize the number of active genes to increase the dynamical variation of the system. At the same time, we do not want GOUT* to grow too big, since the size of GOUT* directly gives the number of genes that are damaged in the asymptotic state. We consider this yield for each of our selection methods, as shown in Fig. 2.4.

Although avoiding connection to high risk genes is a potentially useful feature in preventing connections between damaged genes, we do not want these genes to be kept entirely isolated from the network. In the case of each selection rule, the yield quickly grows as the number of active genes increases, but reaches a maximum when either all genes have been assimilated into the network or the percolation transition of GOUT* has

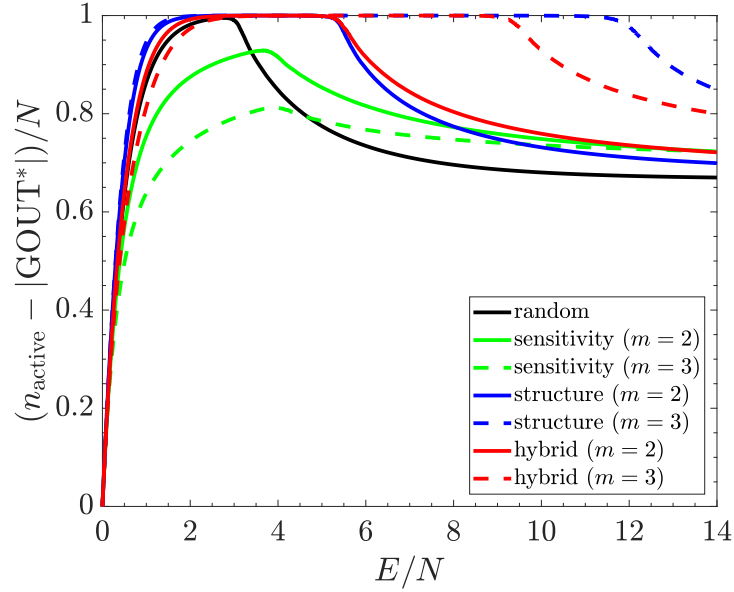


Figure 2.4: The average yield of networks grown under our four selection rules, as described by Eq. (2.5). Results are averaged over 100 runs on networks of size $N = 10^5$. Hybrid and structure-based selection processes show similar yield profiles, consistently outperforming both random and sensitivity-based yields.

been reached. The yield then begins to decrease and approach $1 - N^*/N$, the fraction of undamaged genes in the network.

Sensitivity-based selection is the only of our rules that does not reach a maximum of $Y = 1$ before decreasing, due to a large number of inactive genes. One should note that after peaking, sensitivity-based selection does continue to assimilate genes into the network, but at a slower rate than the growth of $GOUT^*$.

For structure-based and hybrid selection, there is a long period during which the yield is maximized. Both rules are able to quickly incorporate genes into the main network, since small components of size one are much more preferable. Hybrid selection tends to grow slower at first, due to its slight exclusion of high sensitivity genes. For $m = 2$, when both of these rules form $GOUT^*$ at approximately the same time, hybrid

selection performs slightly better, since the slower growth of GOUT* leads to a higher yield after the transition. Increasing to $m = 3$ and beyond, however, the longer delay in percolation makes structure-based selection significantly better.

2.3.4 Correlations

Finally, we consider two forms of correlations in our networks. The first is the in-out degree assortativity, given by

$$r = \frac{\sum_{i \rightarrow j} (d_{\text{in}}^i - \bar{d}_{\text{in}}^s)(d_{\text{out}}^j - \bar{d}_{\text{out}}^t)}{\sqrt{\sum_{i \rightarrow j} (d_{\text{in}}^i - \bar{d}_{\text{in}}^s)^2} \sqrt{\sum_{i \rightarrow j} (d_{\text{out}}^j - \bar{d}_{\text{out}}^t)^2}}, \quad (2.6)$$

where sums are taken over all directed links $i \rightarrow j$, d_{in}^i (d_{out}^j) is the in- (out-) degree of node i (j), and $\bar{d}_{\text{in}}^s$ ($\bar{d}_{\text{out}}^t$) is the average in- (out-) degree of all source (target) nodes averaged over links. The in-out degree assortativity is equivalent to the Pearson correlation coefficient between in-degree of source nodes and out-degree of target nodes, with all averages taken over links. Very positive assortativity indicates a preference of large degree nodes to connect to other large degree nodes, while very negative assortativity, also called high disassortativity, indicates large degree nodes tend to connect to small degree nodes. When the assortativity is near zero (no preference), the network is said to be assortatively neutral.

The second relationship we consider is the correlation between node out-degree and node sensitivity, calculated by the Pearson correlation coefficient between out-degree and sensitivity averaged only over active nodes. Due to the symmetry of our selection rules, the correlation between out-degree and sensitivity is equal to the correlation between in-

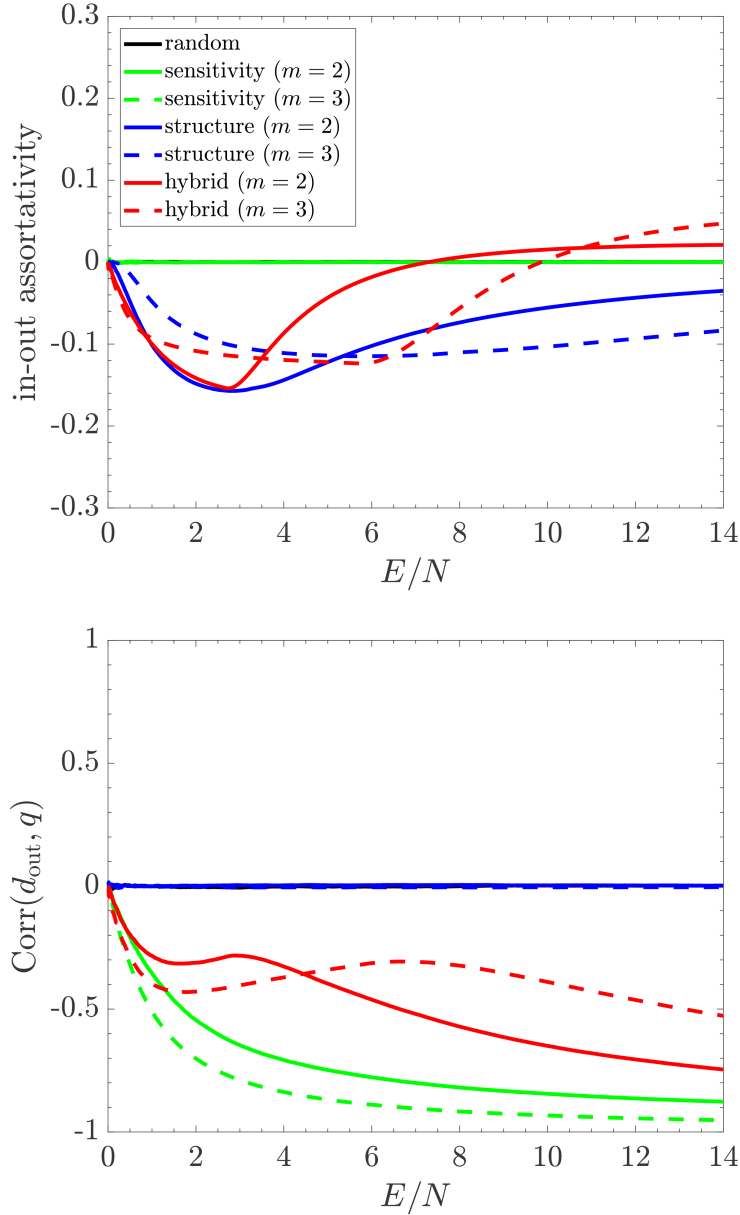


Figure 2.5: In-out assortativity (top) and the degree-sensitivity correlation (bottom) for our four selection rules. Results shown are averages over 100 runs on networks of size $N = 10^5$. While the random and sensitivity-based process are assortatively neutral (i.e., have near zero in-out assortativity), we see that both the structure-based and hybrid selection processes result in disassortative networks before the phase transition, with the former continuing to show weak disassortativity after the transition, while the latter is nearly neutral after the transition. The random and structure-based processes result in no correlation between degree and sensitivity while, as expected, the sensitivity-based selection process creates a negative correlation between a node's out-degree and its sensitivity. The hybrid process shows weaker and non-monotonic negative correlations.

degree and sensitivity. We will refer to either case as the degree-sensitivity correlation, since they are interchangeable. Both the in-out assortativity and degree-sensitivity correlation are shown in Fig. 2.5.

As one should expect, both random selection and sensitivity-based selection are assortatively neutral, since they do not directly consider component sizes or node degree in the decision process. Structure-based selection and hybrid selection are highly disassortative processes, since both will tend to prevent large in-components from connecting to large out-components. Nodes with large in-components typically have large in-degree and similarly with out-components and out-degree. In both of these rules, the assortativity continues to decrease until the critical point at which GSCC percolates, after which the assortativity tends toward zero. Once the giant components are large enough, there is a high probability to select genes from the GSCC. If all m genes are from the GSCC, then they will each have the same component sizes, at which point a link will be chosen either randomly, in the case of structure-based selection, or based on sensitivity, in the case of hybrid selection. Either approach will tend towards neutral assortativity, since the decisions are independent of component size, which relates to degree. Since hybrid selection leads to a faster growing GSCC, its assortativity tends to zero much faster. For the case of $m = 3$, the trend is similar, though the assortativities of structure-based and hybrid selection decrease at a slower rate to a higher (less negative) minimum value. Since higher values of m are better at keeping components small, more components exist near the average component size.

This trend is clarified by considering the degree-sensitivity correlation. First, we note that both random and structure-based selection have uncorrelated degrees and sensi-

tivities, since they ignore sensitivity information. Sensitivity-based selection has a strong negative degree-sensitivity correlation, which is also to be expected since low sensitivity genes have a higher probability of gaining links. The most interesting case is hybrid selection. Initially, when all components are small, sensitivity is a larger influence in hybrid selection, causing a negative degree-sensitivity correlation. Once components are of moderate size, component size begins to take over until the formation of GSCC. As discussed previously, once GSCC forms there is a greater chance that all m potential genes will come from GSCC, in which case all m genes have equal component sizes and sensitivity again takes over.

2.3.5 Connecting with data

We compare our simulated networks to networks reconstructed from data. We consider several reconstructed transcriptional gene regulatory networks for different species, listed in Table 2.1. Using [1], we have estimates on the ages of most genes found in these networks, which allows us to construct an ‘ancestral ordering’ of links. We do this by ordering each gene by age and adding a link once both genes appear. Admittedly, this way of capturing the network through evolutionary time provides only a very coarse view, since it relies on the strong assumption that a regulatory link between two genes appears as soon as the ‘younger’ gene appears. While this assumption is almost certainly violated, we believe it provides a good starting point for connecting with data. For comparison, we also consider a random ordering of the links. In both the ancestral and random orderings, the final networks are identical, with the only difference being the order in which the links

were added.

Due to the small size of the networks, which results because the empirical data is incomplete, the connected components do not have clear transitions. In addition, we do not have known values for the bias or sensitivity of each gene. The sensitivity of a gene may be approximated using expression data, but calculating accurate sensitivities is a challenging problem and beyond the scope of our investigation.

We can, however, analyze the in-out assortativity of the network as a function of the number of links added, as shown in Fig. 2.6. The final values of the in-out assortativity are given in Table 2.1. As we observed in our simulated structure-based and hybrid networks, when we add links using the ancestral ordering that incorporates gene age, we see a general trend in in-out assortativity: a significant early drop to a moderately negative minimum, followed by a slow increase back toward zero. This pattern is clear for 3 of the 4 species for which Ref. [1] provides extensive gene age data: yeast, drosophila, and mouse. We see a similar, yet weaker, trend for human in Fig. 2.6(d), but only after an initial peak with elevated assortativity, which is especially pronounced for the ENCODE data. We believe that the difference in the pattern may be due to the small network size (the ENCODE network includes only 122 genes) and the biases involved in subsampling from the full human gene regulatory network. Nonetheless, the difference between the ancestral ordering and the random ordering for the human data is similar to the other species: the assortativity for the ancestral ordering dips far lower than for the random ordering and stays significantly negative for a much larger range of links added. Perhaps interestingly, the two species with positively assortative final networks, yeast and *E. coli*, are also the only single-celled organisms considered. Unfortunately, incomplete gene age

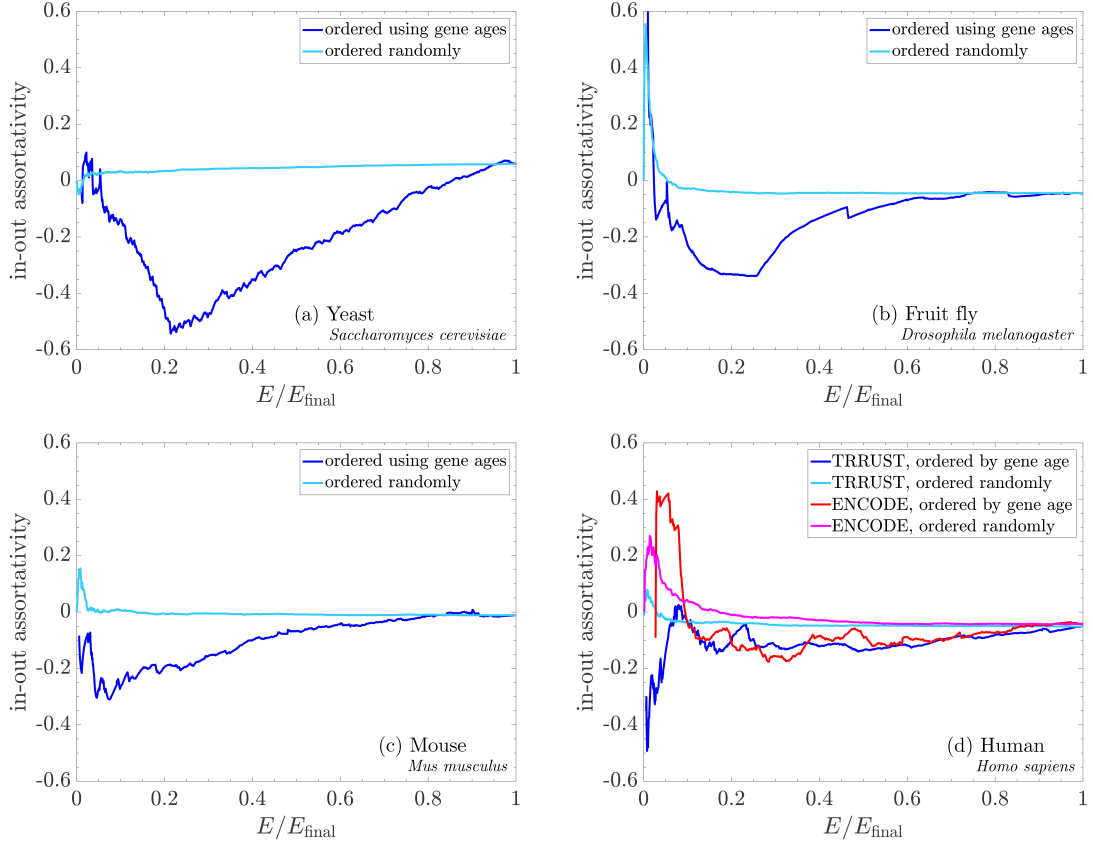


Figure 2.6: In-out assortativity for the transcriptional gene regulatory networks listed in Table 2.1, as a function of the number of links added. Links are ordered using gene age information [1], and a randomized ordering of the links in the final network is shown for comparison. Both orderings yield the same final network. *E. coli* is omitted, due to lack of gene age information from [1]. Note that the horizontal axis is scaled by total number of final links (E_{final}), rather than number of genes (N), to allow for comparison across the different networks.

data from [1] means we cannot verify if the assortativity of the *E. coli* network follows the same trend as the others.

2.4 Discussion

We have seen that using simple competitive rules, a network can be evolved to resist large-scale damage propagation significantly better than the random case. Often this

Species	Nodes	Edges	In-out assort.
<i>E. coli</i> [41]	201	460	0.0475
<i>S. cerevisiae</i> [42]	409	5068	0.0590
<i>D. melanogaster</i> [43]	329	3407	−0.0447
<i>M. musculus</i> [44]	827	2213	−0.010
<i>H. sapiens</i> (TRRUST) [44]	795	1882	−0.0516
<i>H. sapiens</i> (ENCODE) [45]	122	624	−0.0430

Table 2.1: Number of nodes, edges, and the in-out assortativity for the *E. coli*, yeast, fruit fly, mouse, and human transcriptional networks. These in-out assortativity values are also the final values in Fig. 2.6

leads to specialized structure such as higher link density between low sensitivity genes, high disassortativity, and strong negative correlations between degree and sensitivity. Perhaps most surprising is that knowing a gene’s sensitivity is not necessary to effectively delay the onset of a system-wide damaged component, although it does help to ensure that the components remain small. In fact, when we include sensitivity but not structure in the selection rule, more links form between a smaller subset of genes. Not only does this feature result in faster growth of giant components, but it also tends to exclude some high sensitivity genes from contributing to the network at all. Further, the in-out assortativity of our simulated networks evolved using structure-based selection mimics the in-out assortativity of our reconstructed real networks, further corroborating the importance of structure information in the evolution of actual regulatory networks. Future investigations into the connection between this kind of model and real-world data would benefit tremendously from accurate estimates of gene biases.

Surprisingly, using gene sensitivity information along with structure does not provide a benefit, at least when applied through a simple multiplicative rule, when compared to using structure alone. This hybrid rule results in a three phase growth process that

switches between emphasizing low sensitivity genes, then small component genes (which at such a point are high sensitivity), and then back to low sensitivity genes. The drawbacks found in the purely sensitivity-based selection take effect, leading to earlier percolation than if sensitivity were ignored. One can construct more complicated selection rules that incorporate sensitivity and structure to perform better than structure alone, but such rules tend to be overly engineered, and hence we suspect less biological.

One missing aspect in this process is the ability for gene sensitivities to evolve over time. The sensitivity of a gene is an innate dynamical property, and should not be expected to stay constant throughout every generation. In fact, as the structure of the network evolves over time, the dynamical behavior of individual genes would likely change in response. Allowing for the coevolution of structure and sensitivity may lead to a different story, and represents a natural next step.

Data Accessibility

All relevant code and data are publicly available at <https://github.com/bralex1/EvoNets>.

Chapter 3: The sync-contagion model of dual information spread

3.1 Introduction

Many agent-based models have been studied to explore emergent dynamics in populations of interacting individuals. Examples range from modeling the synchronization of flocks of animals [7, 46–48] to the spread of epidemics or information along networks [49–52]. Most of these studies focus on a single type of dynamics, e.g. either the synchronization process or an information cascade, but not both simultaneously. A recent study [47] explored swarming and synchronization at the same time through a ‘swarmalator’ model. The swarmalator model consists of coupled spatial and phase synchronization dynamics. With this model, O’Keeffe, Hong, and Strogatz discovered five type of pattern formation involving clustering in space and phase. In this chapter, we are similarly interested in a combined model, with our focus on the case of synchronization and information cascades coevolving in a population. Like the swarmalators of Ref. [47], the combined ‘sync-contagion’ model has the potential to exhibit new behaviors not otherwise observable in the individual models alone.

Synchronization is a phenomenon in which a group’s behaviors converge over time towards some consensus. Examples include flocking and swarming of animals [7, 46, 53], the songs of Japanese tree frogs [54], humans escaping a fire [55, 56], and the flow of

power in a power grid network [57]. One model of synchronization is the Kuramoto model [58,59], which has been used to analyze the behavior of a network of coupled phase oscillators, but has also been applied to swarming [7, 47, 48] and opinion dynamics [60, 61]. The Kuramoto model is known to exhibit two main types of behavior, based on choice of coupling strength [59, 62, 63]. For coupling above some critical value, the oscillators will synchronize, forming one or more clusters. For coupling below the critical value, the oscillators will instead disperse into an incoherent state, in which the oscillator phases are evenly distributed around a circle.

Contagion models, on the other hand, represent systems where a signal or attribute cascades throughout a population. The archetypal contagion model is the well-studied SIR model of infection spread [49, 64, 65]. In the SIR model, each individual belongs to one of three groups: susceptible, infected, and removed/recovered. The infected individuals spread the infection to susceptible individuals and eventually recover, gaining long-term immunity. Beyond infectious diseases, the SIR model can easily be adapted to model the spread of a computer virus [50, 66] or the spread of information such as news and rumors [51, 52, 67].

When combining the sync and contagion models, we consider a system of individuals where some initial small population is incited by an external stimulus. The presence of this stimulus requires some form of behavioral change of the population represented by a target ‘orientation’ known only to the initial group. The initial group must then send two kinds of information throughout the population: the existence of the triggering stimulus and the target orientation. Specific applications include escape of a group in the presence of a threat, in which the orientation is the literal directional facing of an individual, and

the changing opinions of a population in response to a hot topic, where the orientation is an abstraction of one's 'political orientation'.

A combined sync and contagion model was introduced by Chicoli and Paley [7] to model the shoaling of fish schools and their escape in the presence of a predator. We base our sync-contagion model on the model presented by Chicoli and Paley, but with two key differences. First, the model of Chicoli and Paley only features unidirectional feedback between the two models, where the contagion dynamics influence the sync dynamics but not the other way around. We introduce a bidirectional feedback, to allow the sync system to affect the contagion system, by including an 'acceptance function' that determines the probability of successful contagion spread to an individual based on its orientation. For the second change, we note that Chicoli and Paley investigate the distinction between a naturally coherent and naturally incoherent system through selection of the initial conditions, but in both cases choose system parameters that will lead to synchronization over time. While this is valid for short timescales, we instead choose to distinguish between coherent and incoherent systems by setting the sync model parameters appropriately, which allows us observe the longer-term behaviors for each.

We discover five types of behavior exhibited by our sync-contagion model when considering the success of realignment of the system towards the target orientation. First, if the rate of contagion transmission is not fast enough, then we achieve the trivial scenario of no initial contagion spread. Second, for naturally incoherent systems, successful realign is possible, but only temporary, as the system will return to total incoherence after sufficient time when no contagion is present. Finally, the remaining three behaviors are various levels of permanent align in coherent systems: no/limited realignment, partial

realignment, and full realignment. The distinction between these three behaviors results from the relative strengths of the contagion transmission and the sync coupling and their ability to overcome a ‘barrier’ introduced by the acceptance function.

3.2 Model and methods

In this section, we first present an overview of the two models that we integrate to form our sync-contagion model: the Kuramoto model for synchronization and the SIR model for information cascades. We then describe our unified model that combines the two with bidirectional feedback. Finally, we define our initial conditions and discuss some considerations regarding parameter choices.

3.2.1 Sync model

The Kuramoto model describes the phenomenon of synchronization in a system of coupled oscillators [58, 59] as a function of the coupling strength. While normally a system of differential equations, the Kuramoto model is easily adapted to a discrete-time map model [62, 63, 68] that exhibits similar behavior. The discrete version we use as a foundation most closely resembled the ‘sync map’ presented in [63].

For our sync model, we consider a system of N coupled oscillators. At each time step t , oscillator k updates its orientation θ_t^k according to

$$\theta_{t+1}^k = \theta_t^k + \eta_t^k + \frac{\kappa}{N} \sum_{j=1}^N \sin(\theta_t^j - \theta_t^k), \quad (3.1)$$

where κ is the coupling strength between oscillators and η_t^k is i.i.d. random noise. After

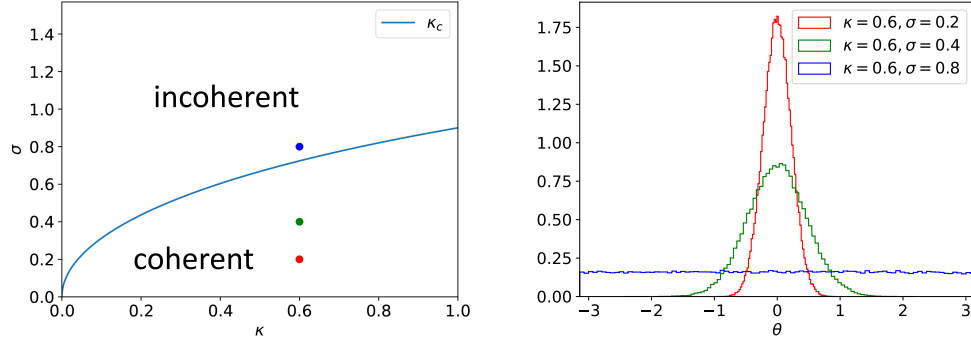


Figure 3.1: **Left:** The critical boundary for the sync model (3.1) with Gaussian noise $\eta_t^k \sim \mathcal{N}(0, \sigma^2)$. Values of (κ, σ) above the blue curve lead to an incoherent system state at equilibrium ($\rho_\infty \approx 0$). Colored dots mark values of (κ, σ) used in the example steady-state distributions shown in the right figure, with colors of the dots matching colors of the distributions. **Right:** Example steady-state distributions for the sync model (3.1) with Gaussian noise $\eta_t^k \sim \mathcal{N}(0, \sigma^2)$ for different values of σ and fixed κ , generated from a simulation of $N = 10^5$ oscillators. Note that each distribution has been shifted to have mean 0. As σ decreases, the steady-state distribution changes from a uniform distribution (the incoherent state) to a symmetric, unimodal distribution (the coherent state), with the width of the distribution decreasing as σ continues to shrink.

each step, all orientations are taken mod 2π and remapped to the interval $[-\pi, \pi)$. For our purposes, the noise can be considered the error in each individual's ability to achieve their desired orientation.

The macroscopic dynamics of the sync model can be captured by the order parameter

$$r_t = \rho_t \exp(i\phi_t) = \frac{1}{N} \sum_{k=1}^N \exp(i\theta_t^k), \quad (3.2)$$

which can be broken down into the *coherence*, $\rho_t = |r_t|$, and the *mean orientation*, $\phi_t = \text{Arg}(r_t)$. As shown in [63], in the large N limit for a specified noise distribution, there is a critical value of $\kappa = \kappa_c$ such that a uniform distribution of oscillator orientations is stable (i.e., $\rho_t \rightarrow 0$) for $\kappa < \kappa_c$ and unstable for $\kappa > \kappa_c$. In particular, for Gaussian noise $\eta_t^k \sim \mathcal{N}(0, \sigma^2)$, the critical value is given by $\kappa_c = 2(\exp(\sigma^2/2) - 1)$, shown in Figure 3.1.

If we restrict ourselves to $\kappa \in [0, 1]$ and finite N , we end up with two classes of behavior, demonstrated in Figure 3.1:

1. For $0 \leq \kappa < \kappa_c$, the stable distribution is approximately uniform on $[-\pi, \pi)$ with $\rho_\infty \approx 0$. We will call such systems *incoherent*.
2. For $\kappa_c < \kappa \leq 1$, the stable distribution is unimodal and symmetric about ϕ_∞ . As κ increases, the width of the distribution decreases, and therefore ρ_∞ increases. We will call these systems *coherent*.

Note that it is possible to use a larger range of κ . We choose a lower bound of $\kappa \geq 0$ so that the system will ‘naturally’ synchronize in the absence of noise, except in the exact case of $\kappa = 0$, where no dynamics occur. For the upper bound, values of $\kappa \geq 2$ can potentially lead to formation of multiple clusters in the steady-state [62]. We are only interested in the single-cluster case, and find that a maximum of $\kappa \leq 1$ is sufficient to illustrate the behaviors of our combined model.

3.2.2 Contagion model

Compartmental models have been the fundamental model for infectious disease spread [64, 65] and other forms of information propagation through a population [examples], with the SIR model being likely the most well-known. In these models, individuals exist in one of several ‘compartments’ (or groups) at each point in time, and transition between the groups according to specified rules. Our contagion model is based on the discrete-time SIR model [65], described in this section.

We consider the case of an infection spreading among a community of N individuals. At any point in time, an individual will belong to exactly one of three groups:

1. *Susceptible* ($\mathcal{S}$) individuals have never been infected, but are capable of becoming infected.
2. *Infected* ($\mathcal{I}$) individuals have the infection and are able to spread it to susceptible individuals.
3. *Removed/Recovered* ($\mathcal{R}$) individuals were previously infected and cannot later become infected.

We begin with everyone in $\mathcal{S}$ at time $t = 0$, except for a few initial infected individuals, and as time progresses individuals can transition between groups according to two rules. First, a susceptible individual can become infected upon being exposed to an infected individual. If we assume a fully-mixed population, where the transmission coefficient β is defined as the contact rate $\times$ the transmission probability. For a fully mixed system, a susceptible individual at time t becomes infected with probability $\beta|\mathcal{I}_t|/N$, where $|\mathcal{I}_t|$ denotes the size of the infected population at time t . Second, any infected individual at time t becomes removed (or recovers) with probability γ . This process is summarized in Figure 3.2. The mean behavior of the contagion model can be described by the system of equations

$$\begin{aligned} S_{t+1} &= S_t(1 - \beta I_t), \\ I_{t+1} &= I_t(1 + \beta S_t - \gamma), \\ R_{t+1} &= R_t + \gamma I_t, \end{aligned} \tag{3.3}$$

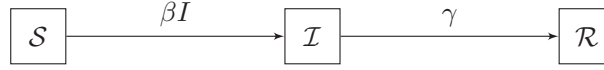


Figure 3.2: Transition diagram for the contagion model.

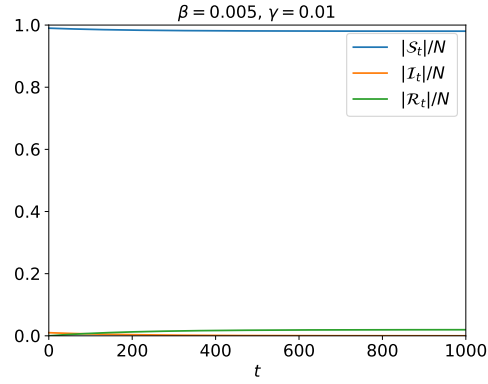
where $S_t = |\mathcal{S}_t|/N$, $I_t = |\mathcal{I}_t|/N$, and $R_t = |\mathcal{R}_t|/N$.

The dynamics of the contagion model are fully determined by the *transmission coefficient* β , the *recovery rate* γ , and the initial infection probability p_0^I . In the case of $I_0 \approx 0$, we have the restriction that $\beta/\gamma > 1$ is required for the initial infection to grow. Examples of the contagion dynamics over time for different β and γ are shown in Figure 3.3.

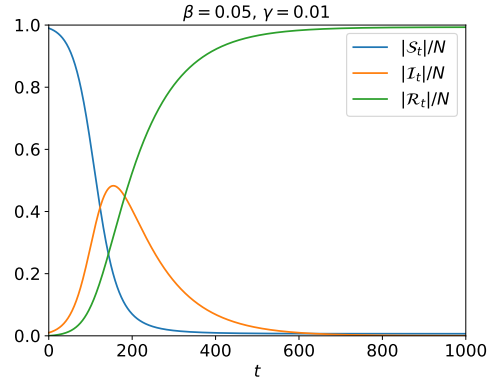
3.2.3 Combining the models: the sync-contagion model

To combine our sync and contagion models, we again consider a system of N individuals, each with an orientation θ_t^k , but also belonging to one of $\mathcal{S}$, $\mathcal{I}$, or $\mathcal{R}$ at each time t . We imagine that a few individuals in the population are exposed to special information and become ‘infected’ by setting their orientation to the target orientation reflected by this special information, with all other individuals starting in the susceptible state. For schools of fish, the target orientation might correspond to the direction of escape from a predator. Susceptible individuals become infected based on interacting with infected individuals as well as the distance between their orientation and the target orientation. If a susceptible individual’s orientation differs too greatly from the the target orientation, then the probability of transmission is reduced.

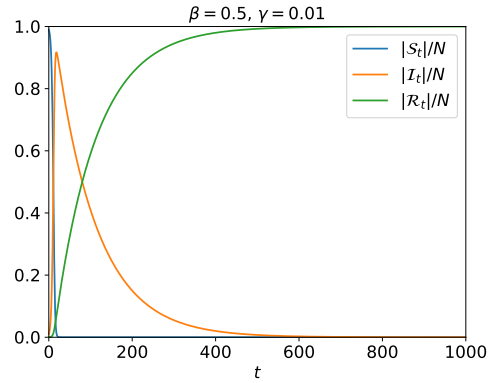
Low transmission, moderate recovery



Moderate transmission, moderate recovery



High transmission, moderate recovery



Moderate transmission, low recovery

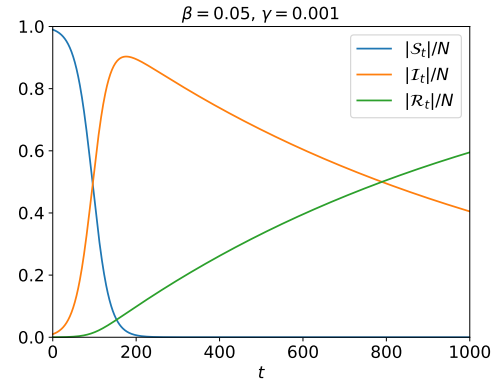


Figure 3.3: Example outcomes of the contagion model for different choices of transmission coefficient β and recovery rate γ . In general, larger values of β lead to greater exponential growth of the infected population (and decay of the susceptible population), while larger values of γ lead to faster exponential decay of the infected population (and growth of the removed population). In the case where $\beta/\gamma < 1$ (top left), the infection immediately dies out.

Specifically, we modify Eq. (3.1) to depend on an individual's infection status as

$$\theta_{t+1}^k = \begin{cases} \theta_t^k + \eta_t^k + \frac{\kappa}{N} \sum_{j=1}^N \sin(\theta_t^j - \theta_t^k), & \text{if } k \in \mathcal{S}_{t+1} \cup \mathcal{R}_{t+1}, \\ \phi_t^{\mathcal{I}} + \eta_t^k, & \text{if } k \in \mathcal{I}_{t+1}, \end{cases} \quad (3.4)$$

with

$$\phi_t^{\mathcal{I}} = \text{Arg} \left[\frac{1}{|\mathcal{I}_t|} \sum_{k \in \mathcal{I}_t} \exp(i\theta_t^k) \right]. \quad (3.5)$$

In other words, individuals that are susceptible or removed follow the normal sync model rules, while individuals that are infected align to the mean orientation of the infected subpopulation, calculated from the previous time step.

The bidirectional feedback is achieved by making the transition probabilities between the contagion groups to depend on an individual's orientation. Specifically, upon exposing a susceptible individual to the infection, there is a probability $\alpha(\theta)$ that the infection takes hold and a $1 - \alpha(\theta)$ probability that the individual instead becomes immediately removed. The *acceptance function* $\alpha(\theta)$ depends on the distance between the individual's orientation θ and the target orientation $\theta_{\odot}$, with infection success increasing as this distance decreases. The new transition diagram for the sync-contagion model is then given in Figure 3.4. The resulting mean behavior is summarized by

$$\begin{aligned} S_{t+1} &= S_t(1 - \beta I_t), \\ I_{t+1} &= I_t(1 + \beta S_t \langle \alpha(\theta_t^k) \rangle_k - \gamma), \\ R_{t+1} &= R_t + I_t(\beta S_t(1 - \langle \alpha(\theta_t^k) \rangle_k) + \gamma). \end{aligned} \quad (3.6)$$

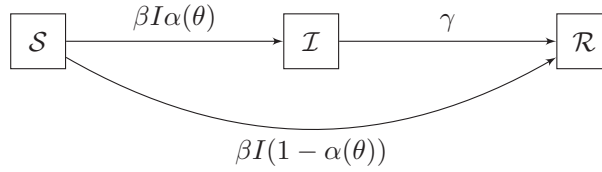


Figure 3.4: Transition diagram for the sync-contagion model. We have updated the transition diagram from Figure 3.2 with the effect of the acceptance function $\alpha(\theta)$.

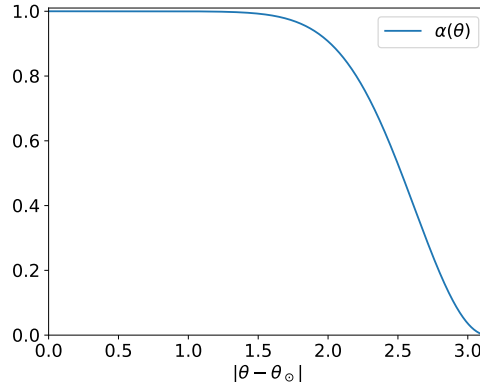


Figure 3.5: Plot of the acceptance function $\alpha(\theta)$ used in our simulations. Individuals with orientations near the target orientation $\theta_\odot$ are more likely to take on the infection when exposed.

We choose $\alpha(\theta)$ to be defined as

$$\alpha(\theta) = 1 - \exp\left(-\frac{(|\theta - \theta_\odot| - \pi)^2}{2(\pi/6)^2}\right), \quad (3.7)$$

as shown in Figure 3.5. We note that the specific choice of $\alpha(\theta)$ is not too important, but Eq. (3.7) has key features of $\alpha(\theta)$ strictly increases as $|\theta - \theta_\odot|$ decreases, $\alpha(\theta_\odot) \approx 1$, and $\alpha(\theta_\odot + \pi) = 0$. Other choices of $\alpha(\theta)$ with these features will produce similar qualitative results.

We define $t = 0$ as the time when the first infection is introduced into the system. Before this point, we allow the orientations to equilibrate according to the stable

distribution determined by Eq. (3.1), and then redefine our coordinates so that the mean orientation at time $t = 0$ is set to $\phi_0 = \pi$.

We introduce the infection at $t = 0$ by setting each individual as infected with probability p_0^I , leading to $I_0 \approx p_0^I$. If an individual k is infected at $t = 0$ (i.e., $k \in \mathcal{I}_0$), then we immediately reorient $\theta_0^k = \theta_\odot + \eta_0^k$. For our analysis, we choose $\theta_\odot = 0$, causing the initial susceptible and infected populations to be as far apart as possible and presenting a ‘worst-case’ scenario for the system.

3.2.4 Applications

Our sync-contagion model can be used to represent systems with two pieces of information to spread: awareness of an external stimulus (the ‘infection’) and the initial response orientation, $\theta_\odot$, for the small set of individuals infected at $t = 0$. Infected individuals are the primary drivers in realignment of the system towards $\theta_\odot$. Note that only the original infected population ‘knows’ to redirect towards $\theta_\odot$, but this knowledge is preserved within the infected population since Eq. (3.4) leads to $\phi_t^{\mathcal{I}} \approx \theta_\odot$ as long as $|\mathcal{I}_t| > 0$. Those who are uninfected may align towards $\theta_\odot$ either by becoming infected themselves, or by the base synchronization dynamics determined by Eq. (3.1).

Applying the model for escape in the presence of the threat, similar to Ref. [7], each individual moves in some direction θ . Knowledge of the threat is propagated as a startle, which eventually dies down, while the desired escape direction is $\theta_\odot$. Individuals who are moving in a direction too different from $\theta_\odot$ do not react their startled neighbors, either due to not seeing others moving in the opposite direction or because the change in

direction would be too large to be feasible. This non-reaction to the startle is captured through the acceptance function $\alpha(\theta)$.

As a model of opinion dynamics, the orientation may represent an individual's stance on a particular topic. The infection then represents interest in some current 'hot topic' that might redirect an individual towards a new stance $\theta_\odot$. Individuals who take interest become promoters of the stance $\theta_\odot$ and try to encourage others to become promoters. Eventually, interest in the hot topic dissipates. Individuals whose stances are too distinct would refuse to become promoters upon learning of the hot topic, as captured by $\alpha(\theta)$.

Take, for instance, θ to be one's opinion on social injustice and the existence of systemic racism. The killings of George Floyd, Breonna Taylor, Ahmaud Arbery, and countless others triggered a national dialogue on the topic, with many taking part in the Black Lives Matter protests. The protesters can be seen as promoters of the stance that systemic racism is real and is an issue that needs to be addressed. Many individuals were convinced by the movement and joined the side of the promoters, further spreading the discussion. However, many individuals were not swayed, due to their political stance deviating too greatly from the intended position $\theta_\odot$ of the promoters, an effect captured by $\alpha(\theta)$ in our model.

3.2.5 Parameter restrictions

For our combined sync-contagion system, we have five parameters: coupling strength κ , noise distribution width σ , transmission coefficient β , recovery rate γ , and initial infec-

tion probability p_0^I . We can reduce this to three by first observing that realignment of ϕ_t is primarily driven by the infected population. The susceptible and removed populations can only passively realign according to the sync model (3.1), with more weight given towards $\theta_\odot$ when $|\mathcal{I}_t|$ is larger. The two parameters that directly affect $|\mathcal{I}_t|$ are p_0^I , for which larger values give the infected population a greater head start, and γ , for which smaller values allow the size of the infected population to be sustained for longer. Since the effects of these two parameters are fairly straightforward, we will set them to reasonable values of $p_0^I = 0.01$ and $\gamma = 0.01$.

The effects of κ , σ , and β are less obvious. However, we can note that all three play a role in the requirement for the growth of the initial infection. For $|\mathcal{I}_0| \approx 0$, the initial infection will grow on average only when $\beta\alpha_0/\gamma > 1$, where

$$\alpha_0 = \frac{1}{N} \sum_{k=1}^N \alpha(\theta_0^k). \quad (3.8)$$

Recall that the distribution of θ_0^k depends on the stable distribution for Eq. (3.1), which in turn depends on κ and σ .

3.2.6 Measures

We need to evaluate the emergent dynamics of our model in two ways. First, we need to determine the success of the sync model in realigning the system towards the target direction $\theta_\odot$. Second, we need to determine how well infection information is spread through the contagion model. We describe our means of measuring success in this section.

3.2.6.1 Coherence, mean orientation, and realignment

Just as with the original sync model, we can consider the order parameter r_t given in Eq. (3.2), which gives the coherence of the orientation distribution $\rho_t = |r_t|$ and the mean orientation $\phi_t = \text{Arg}(r_t)$. Furthermore, we can consider the coherence and mean orientation of the subpopulations $\mathcal{S}$, $\mathcal{I}$, and $\mathcal{R}$ by averaging over only members of those groups, as was done for $\phi_t^{\mathcal{I}}$ in Eq. (3.5).

We consider realignment to be successful if ϕ_∞ becomes reasonably close to $\theta_\odot$. Since the infection will always die out and the system will always return to the basic sync model (3.1), permanent realignment is only possible in coherent systems.

For incoherent systems, only temporary (re)alignment is possible, since the system will naturally return to the totally incoherent state. In this case, the order parameter ρ_t is more useful, since it will generally increase only when a cluster forms around $\theta_\odot$, with strongest alignment likely occurring when ρ_t is maximized.

3.2.6.2 Maximum infection fraction

We can also consider the maximum infected fraction

$$I_{\max} = \max_t I_t. \quad (3.9)$$

The value of $I_{\max}$ is especially meaningful in the context of opinion dynamics, where the size of the infected population represents the number of concurrent promoters of the stance $\theta_\odot$. Even if realignment did not occur, maximizing the ‘visibility’ of a hot topic is

still desirable.

Furthermore, for incoherent systems, if we assume the sync model dynamics are fast enough such that uninfected individuals are always uniformly distributed, then in the large N limit, ρ_t is given by

$$\rho_t = I_t \rho_t^{\mathcal{I}}. \quad (3.10)$$

Note that for all t and large N that $\theta_t^k \sim \mathcal{N}(0, \sigma^2)$ for $k \in \mathcal{I}_t$, meaning $\rho_t^{\mathcal{I}} \equiv \rho_0^{\mathcal{I}}$ is constant, and ρ_t is then maximized precisely when I_t is maximized. Thus, $I_{\max}$ will directly give the alignment success for an incoherent system, removing the need to consider $\rho_{\max}$ at all.

We find that this assumption requires σ to be large enough for the choice of κ , since increased noise leads to faster convergence to the incoherent state. In cases where this approximation is invalid, however, $\rho_{\max}$ still occurs approximately simultaneously with $I_{\max}$ in incoherent systems.

3.2.6.3 Maximum cosine mean

While the final mean orientation ϕ_{∞} works well to evaluate coherent systems, and $\rho_{\max}$ (or $I_{\max}$) works well for incoherent systems, it is desirable to have a measure to compare coherent to incoherent systems. One measure that works in both cases is the maximum cosine mean

$$C_{\max} = \max_t \frac{1}{N} \sum_{k=1}^N \cos(\theta_t^k) = \max_t \rho_t \cos(\phi_t). \quad (3.11)$$

The value $C_{\max}$ is larger when ϕ_t is close to our target orientation $\theta_{\odot} = 0$, but penalized for wider distributions of orientations (ρ_t small).

In incoherent systems, symmetry causes $\phi_t = \theta_{\odot}$ as long as $|\mathcal{I}_t| > 0$, so that $C_{\max} = \rho_{\max}$. Meanwhile for coherent systems, $C_{\max}$ will depend more on the final orientation ϕ_{∞} . Thus, $C_{\max}$ gives a way to compare the primary measure of coherent systems (ϕ_{∞}) to the measure for incoherent systems ($\rho_{\max}$).

3.3 Results

In this section, we present the key results of the sync-contagion model. We begin by observing the general behavior of the system in the incoherent and coherent cases, and then consider the effects of modifying our three remaining parameters: noise distribution width σ , coupling strength κ , and transmission coefficient β .

3.3.1 General behavior

To begin our investigation, we take a look at a sample of the sync-contagion dynamics applied to both an incoherent and a coherent system.

As described in Section 3.2.6.2, if we assume that the orientations of uninfected individuals are always uniformly distributed in an incoherent system, then the coherence ρ_t will be directly proportional to $|\mathcal{I}_t|$. In the incoherent case shown in Figure 3.6, this assumption does not exactly hold, since $\rho_t^{S \cup R}$ becomes noticeably greater than zero; however, comparing the graphs of $\rho_t^{\mathcal{I}}$ and $|\mathcal{I}|/N$ still shows a close relationship regardless. As long as the infection exists, the mean orientation remains near $\theta_{\odot}$, and ρ_t becomes more

positive with the larger infection. However, once the infection dies out, the system returns to the incoherent state, leading to the mean orientation becoming a random value and ρ_t approximately zero.

For coherent systems, the rise of the infection actually causes a decrease in coherence, shown in Figure 3.6, as individuals are split between one distribution centered on the mean orientation $\phi_t^{\mathcal{I}}$ of the infected population and another distribution centered on the mean orientation $\phi_t^{S \cup \mathcal{R}}$ of uninfected population. Over time, $\phi_t^{S \cup \mathcal{R}}$ is pulled towards $\phi_t^{\mathcal{I}}$, leading to permanent realignment of the system, ideally keeping ϕ_∞ near $\theta_\odot$ after the infection ends.

In both cases, since $\rho_t^{\mathcal{I}}$ is essentially constant, the change in ρ_t follows the change in $\rho_t^{S \cup \mathcal{R}}$. Note that as the infection dies out, the coherence of the infected subpopulation becomes more noisy, as fewer members remain.

3.3.2 Modifying the width σ of the noise distribution

We determined the effects of the recovery rate γ and initial infection probability p_0^I in Section 3.2.5. Of our three remaining parameters, the standard deviation σ of the noise term η_t^k has the most straightforward effect. First, recall that for fixed κ , the value of σ will establish whether the system is coherent or incoherent. Beyond this effect, the value of σ that best propagates the infection and realignment actually depends on which of these two synchronization tendencies the system has. By comparison, ideal values γ and p_0^I are independent of synchronization tendency.

For incoherent systems, the ‘goal’ of the infection is to maximize ρ_t and $I_{\max}$. If we

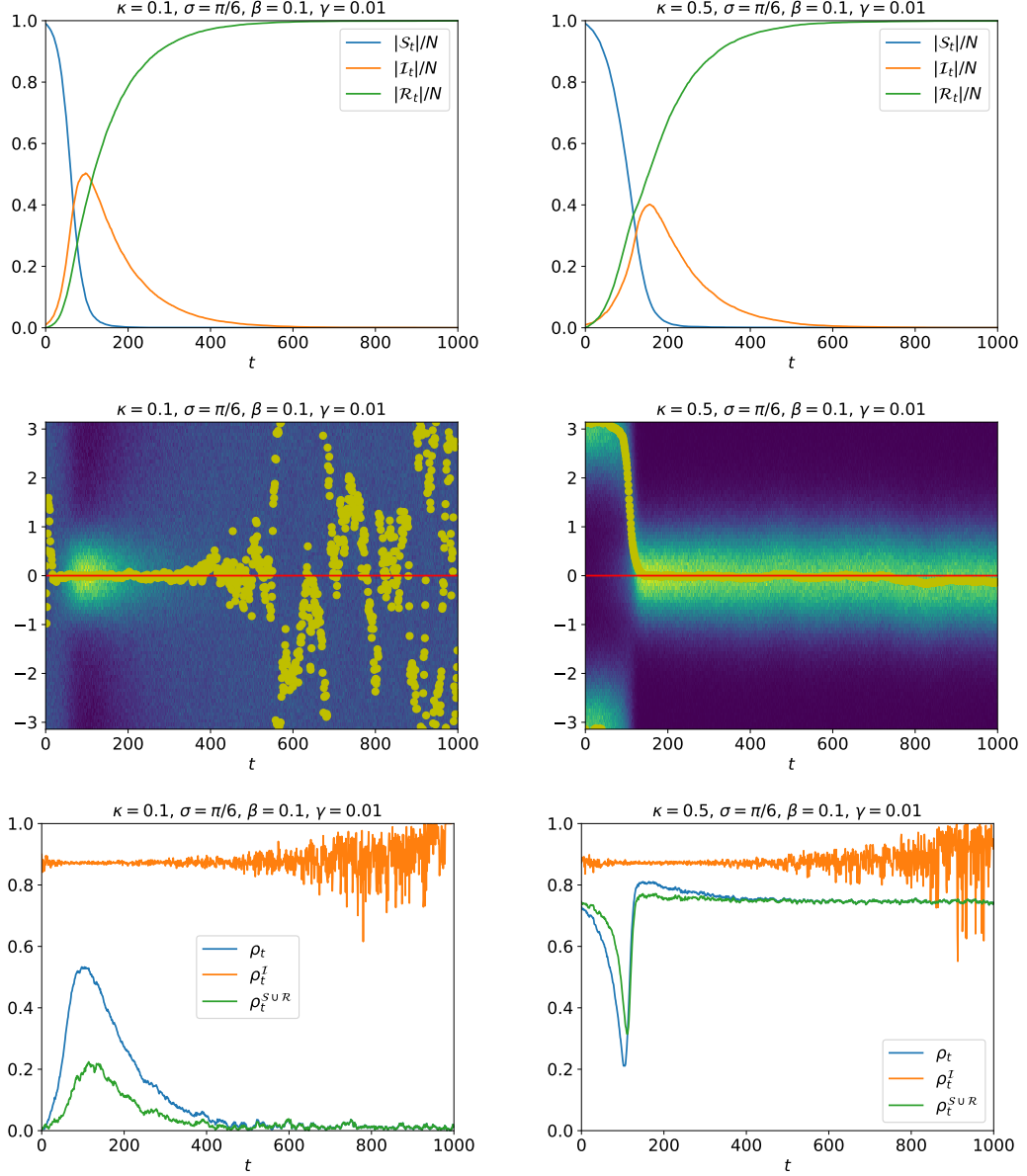


Figure 3.6: Example behavior of an incoherent (left) and coherent (right) system. Simulations used a population of $N = 10^4$ oscillators. **Top row:** Fractions of the population in each category (susceptible, infected, and removed) over time. **Middle row:** Distribution of orientations over time. The red center line indicates $\theta_\odot = 0$. The mean angle ϕ_t is marked by a yellow dot. **Bottom row:** The coherence of the full population ρ_t , the infected subpopulation ρ_t^I , and the uninfected subpopulation $\rho_t^{S \cup R}$.

again assume that uninfected individuals in an incoherent system always have uniformly distributed orientations, then $\langle \alpha(\theta_t^k) \rangle_k = \alpha_0$ for all t . The average probability of a susceptible becoming infected is then $\beta \alpha_0 |\mathcal{I}_t|/N$, where α_0 is the mean of $\alpha(\theta)$ over the uniform distribution on $[-\pi, \pi)$. Hence, the contagion dynamics, particularly $I_{\max}$, are independent of σ .

We do note, however, that

$$\rho_t \approx \frac{|\mathcal{I}_t|}{N} \rho_0^{\mathcal{I}} = \frac{|\mathcal{I}_t|}{N} \exp(-\sigma^2/2), \quad (3.12)$$

which is dependent on σ and decreases as σ increases. We can conclude that lower values of σ are therefore beneficial to incoherent systems, as it allows the distribution of orientations of infected individuals to be tighter around $\theta_{\odot}$.

By comparison, coherent systems tend to form one cluster of infected individuals oriented around $\phi_t^{\mathcal{I}}$ near $\theta_{\odot} = 0$ and one cluster of uninfected individuals around ϕ_t^{SUR} , initially near π , the initial mean orientation. The infection needs sufficient susceptible individuals to be near $\theta_{\odot}$ for the infection to continue, due to $\alpha(\theta)$. If too many individuals are far from $\theta_{\odot}$, then the infection will end and soon after realignment will end. Smaller values of σ are therefore detrimental to coherent systems by causing a tighter initial distribution of susceptible individuals around $\theta = \pi$, making it harder to sustain the infection.

Seen another way, larger σ , and therefore a wider distribution of susceptible individuals' orientations, will speed up the rate of realignment in coherent systems, as shown in Figure 3.7. This wider distribution allows $\alpha(\theta)$ to capture more susceptible individuals,

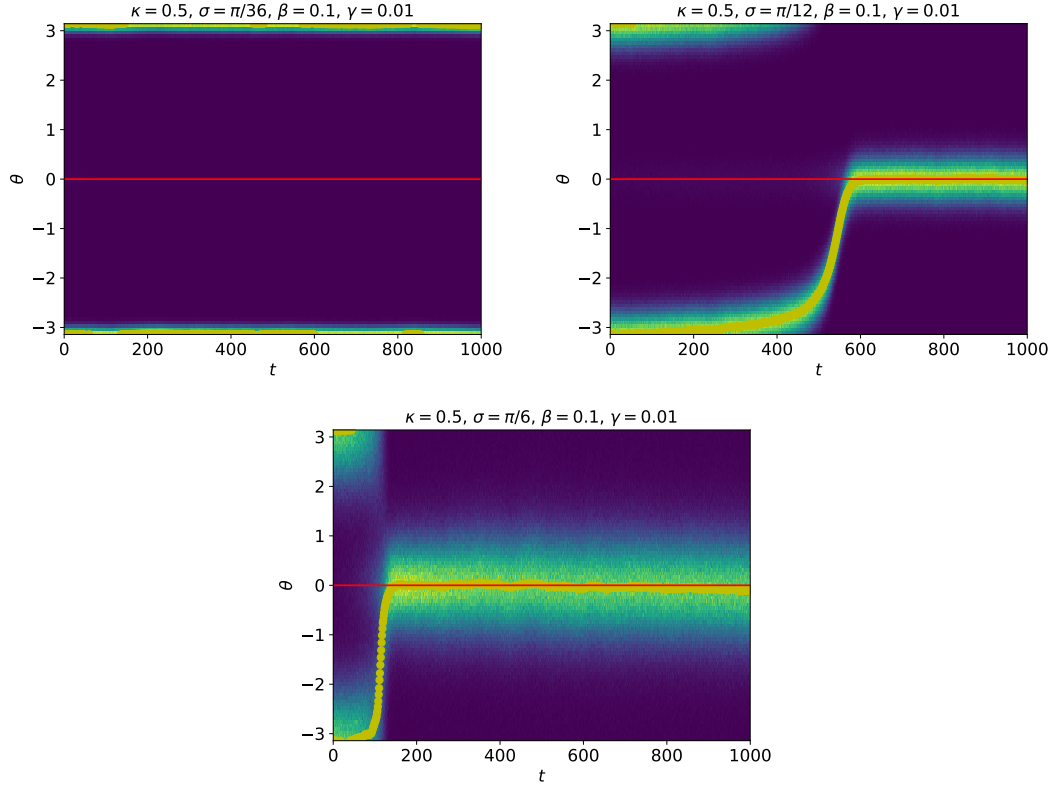


Figure 3.7: Distributions of orientations over time for coherent systems with different choices of noise distribution width σ and $\kappa = 0.5$, $\beta = 0.1$, and $\gamma = 0.01$. The mean angle ϕ_t is marked by yellow dots, and the central red line indicates $\theta_{\odot} = 0$. As the width of the noise distribution increases, the uninfected population realigns towards $\theta_{\odot}$ at a faster rate, at the cost of a less tight distribution.

fueling the infection, and consequently creating more weight towards $\phi_t^{\mathcal{I}}$.

3.3.3 Modifying the coupling strength κ

We next consider the effect of the coupling strength κ . For fixed σ , the value of κ determines whether the system is coherent or incoherent. For incoherent systems, the coupling strength κ is largely unimportant, other than the requirement $\kappa < \kappa_c$. The dynamics are driven primarily by σ , as just discussed, and β , as we will see later.

In coherent systems, κ serves two roles. First, similar to the effect of small σ , larger

κ in coherent systems will lead to a tighter distribution of uninfected orientations around $\phi_t^{S\cup R}$, thus potentially preventing a spread of infection due to $\alpha(\theta)$. However, stronger coupling will also lead to a faster realignment rate of $\phi_t^{S\cup R}$ towards ϕ_t^I . The balance of these two features is key in successful realignment.

We demonstrate the effect of adjusting both κ and β in Figure 3.8, and show a slice for constant β in Figure 3.9. We can see that, indeed, incoherent systems behave independently of κ (for $\kappa < \kappa_c$, while coherent systems exhibit more complex behavior.

For κ slightly above the critical value, where the stable distribution for Eq. (3.1) is quite wide, the infection is easily sustained and ϕ_t realigns towards $\theta_\odot$. As κ increases, the stable distribution tightens up, making infection harder to sustain. In order for the realignment speed to move the mean orientation towards $\theta_\odot$ within an acceptable range for $\alpha(\theta)$ before the infection dies out, κ must be increased further. Note that once $\phi_t^{S\cup R}$ is within a certain proximity of $\theta_\odot$ that $\alpha(\theta) \approx 1$ for all individuals, essentially guaranteeing successful realignment. As previously discussed, larger σ benefits realignment, to the point where κ does not matter.

The regions for realignment success also match up generally well with the higher values of $I_{\max}$ and $C_{\max}$. In the case of large σ , where realignment is guaranteed for coherent systems, the values of $I_{\max}$ are generally higher, but do still exhibit the ‘dip’ for intermediate values of κ . For $C_{\max}$ in high σ coherent systems, larger κ is more beneficial. When realignment is guaranteed, the value of ρ_t is more influential on $C_{\max}$, which is higher for larger κ .

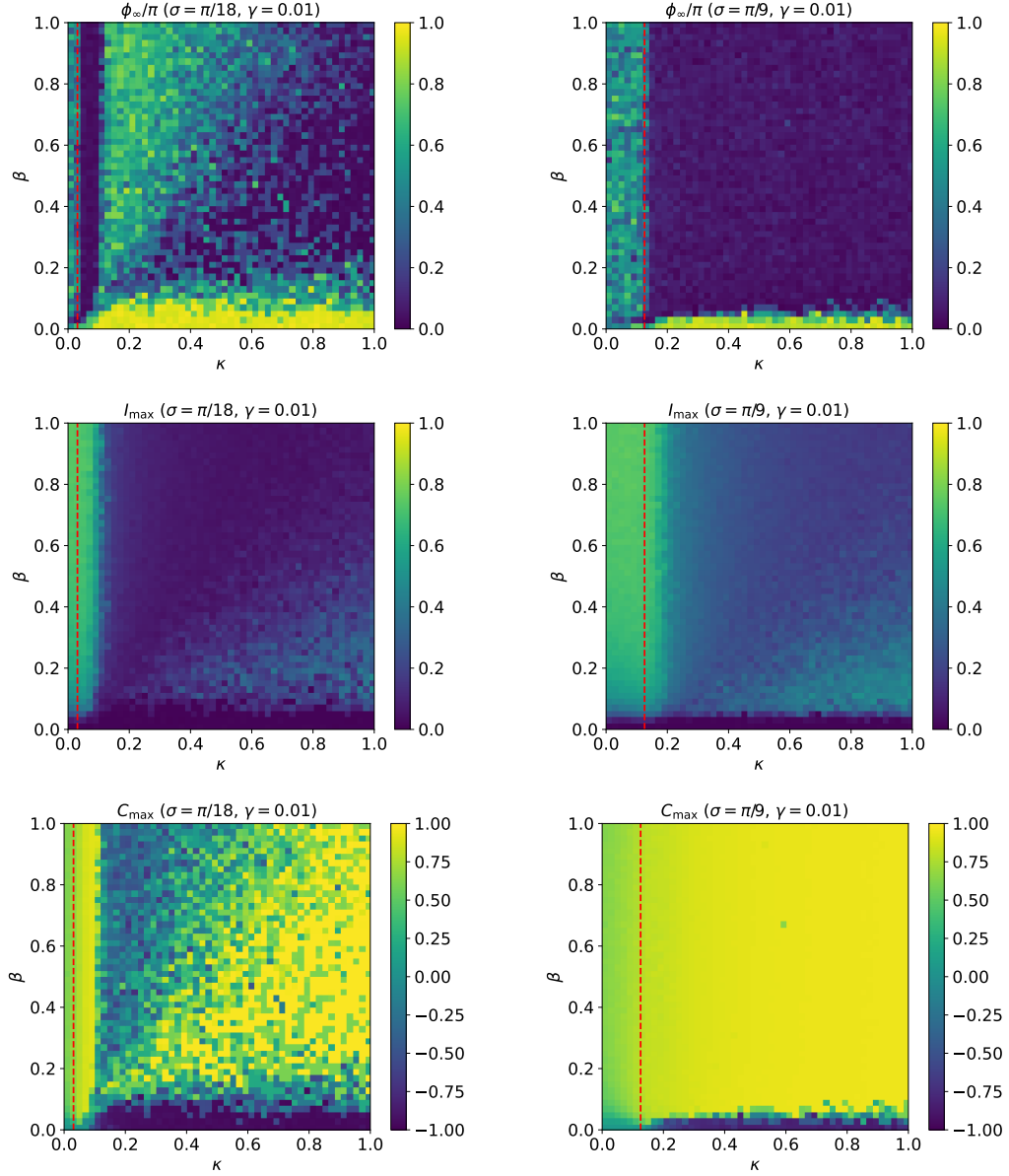


Figure 3.8: Heatmaps of our measures in $\kappa\beta$ -space for low noise ($\sigma = \pi/18$, left) and moderate noise ($\sigma = \pi/9$, right). Results are averaged over five realizations each with a system of $N = 1000$ oscillators. The vertical dashed red line indicates the critical value of κ_c for each value of σ . Incoherent systems exist to the left of the red line, while coherent systems exist to the right. **Top row:** Absolute value of the final mean angle ϕ_∞ scaled by π . For incoherent systems, the final mean angle is a random value, leading to a mean near 0.5. **Middle row:** The maximum size of the infected population $I_{\max}$ as a fraction of the total population. **Bottom row:** The maximum mean cosine $C_{\max}$, which takes into account both the mean orientation ϕ_t and the coherence ρ_t of the distribution.

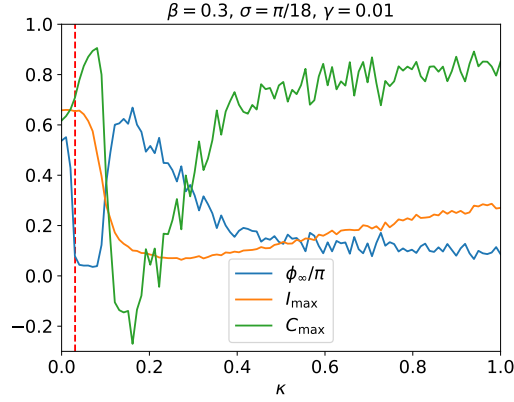


Figure 3.9: Our measures for fixed values of $\beta = 0.3$, $\sigma = \pi/18$, and $\gamma = 0.01$ as κ varies. The vertical dashed red line indicates the critical value of κ_c for $\sigma = \pi/18$. For coherent systems ($\kappa > \kappa_c$), intermediate values of β lead to a drop in performance according to all three measures. Within this region, the distribution is not wide enough (κ too high) and does not realign fast enough (κ too low) to be within an appropriate region for the infection to continue, as required by the acceptance function $\alpha(\theta)$.

3.3.4 Modifying the transmission coefficient β

Finally, we study the effects of adjusting the transmission coefficient β . We again see that the contribution of $\alpha(\theta)$ on the dynamics is key to understanding the effect of β .

Recall that temporary alignment in incoherent systems is stronger when the infected population is larger. To grow the infection as large as possible, we simply need to grow it as fast as possible. If we again assume that orientations of uninfected individuals are uniformly distributed, then we can ignore the effects of $\alpha(\theta)$ and increase the infection rate by increasing β . This effect is clearly shown in Figure 3.10, where larger values of β directly lead to an increase in $I_{\max}$ and $C_{\max}$.

The dynamics when considering the effect of β in coherent systems is similar to our considerations for κ . By increasing β , we increase the transmission coefficient between infected and susceptible individuals. To sustain the infection, we require β to be

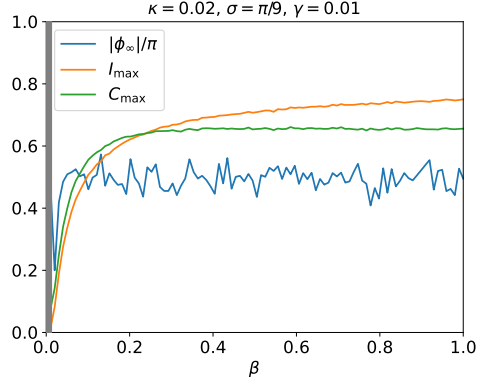


Figure 3.10: Our three measures for different varying values of β on incoherent systems with $\kappa = 0.02$, $\gamma = 0.01$, and $\sigma = \pi/9$. Results are simulated on a system of $N = 1000$ oscillators and averaged over 100 realizations. The large grey box marks the region where $\beta\alpha_o/\gamma < 1$, where the initial infection does not spread. The final mean orientation in an incoherent system is a random value, which leads to $|\phi_\infty|/\pi$ averaging out to 0.5 for all values of β . The maximum size of the infected population and the maximum mean cosine measure both increase as the transmission coefficient increases.

sufficiently large. However, because of $\alpha(\theta)$, if the distribution of susceptible individuals is too far from $\theta_\odot$ during an exposure, then each exposure will likely lead to a transition from susceptible to removed instead. Consequently, it is necessary for the transmission coefficient to be high enough to prevent the infection from dying out (due to recoveries), but low enough to give time for partial realignment of the susceptible orientations to get in proper range of $\theta_\odot$ for $\alpha(\theta)$. We can see a comparison of a successful and failed realignment in Figure 3.11, where the failure is due to increasing the transmission coefficient.

Increasing κ speeds up the sync dynamics, while increasing β speeds up the contagion dynamics. As seen in Figure 3.8, in order for the system to achieve realignment to the target direction, increasing β requires increasing κ so as to preserve the careful balance.

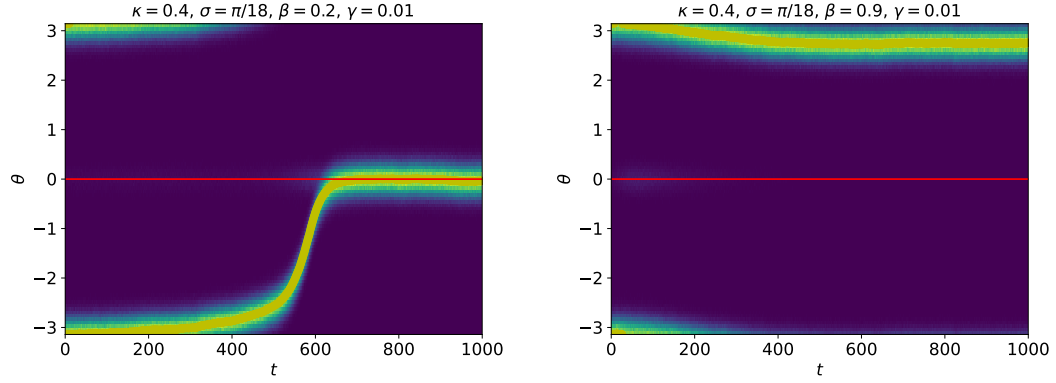


Figure 3.11: Distributions of orientation over time for $\beta = 0.2$ (left) and $\beta = 0.9$ (right) for two coherent systems with $\kappa = 0.4$, $\sigma = \pi/18$ and $\gamma = 0.01$. The mean orientation ϕ_t is indicated by yellow dots, and the horizontal red line indicates $\theta_{\odot} = 0$. When β is made too large, the exposures occur too quickly, leading to most exposures turning into instant removals, due to $\alpha(\theta)$. The transmission coefficient must be slow enough to allow for partial realignment of the uninfected population before the majority of exposures occur, while still large enough to sustain the infection at low levels.

3.3.5 Behavior classification

In total, we can classify the behavior of the sync-contagion model into five categories: (a) infectionless, where $\beta\alpha_0/\gamma < 1$, leading to no initial infection; (b) temporary realignment, which includes all incoherent systems for which the initial infection spreads, causing ϕ_t to temporarily align with $\theta_{\odot}$ before the system returns to an incoherent state; (c) full permanent realignment, where ϕ_{∞} is close to $\theta_{\odot}$; (d) partial permanent realignment, where ϕ_t shifts away from its initial orientation, but ϕ_{∞} is not close to $\theta_{\odot}$; and (e) limited/no realignment, where ϕ_t does not move far, if at all. Permanent realignment only occur in coherent systems. In Figure 3.12, we indicate example regions in $\kappa\beta$ -space corresponding to these five classifications.

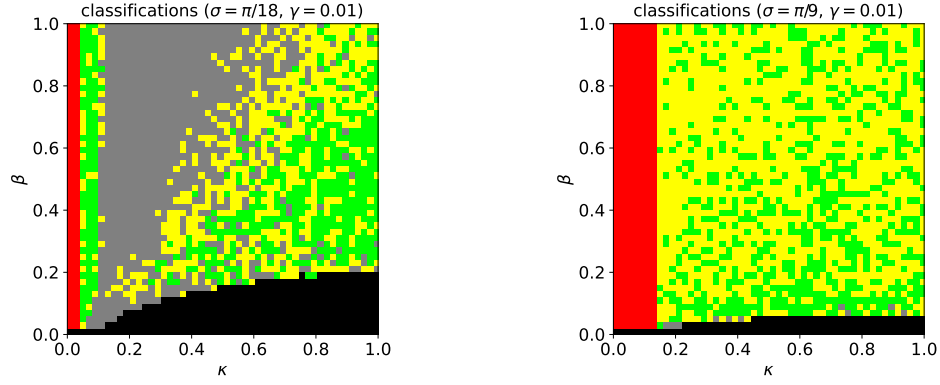


Figure 3.12: Classifications of sync-contagion model behaviors, experimentally derived from the results shown in Figure 3.8. The classification regions are as follows: infectionless (black), temporary realignment (red), full permanent realignment (green), partial permanent realignment (yellow), and limited/no realignment (grey). Partial permanent realignment is based on ϕ_∞ being within $\pi/6$ of the target orientation. Similarly, limited/no realignment is based on ϕ_∞ being further than $5\pi/6$ from the target orientation.

3.4 Discussion

We have introduced a simple model for information propagation that combines the sync dynamics of a Kuramoto-like model with the cascading dynamics of a contagion model. Distinct from other models, we have introduced a bidirectional feedback between the two systems, where the sync dynamics affect the contagion dynamics, which in turn affect the sync dynamics.

Summarizing our findings, we first note the effects of the sync model parameters, coupling strength κ and noise distribution width σ , on the contagion dynamics. For fixed σ , the choice of κ relative to a critical value κ_c will determine whether the sync model exhibits coherent or incoherent behavior. These two behaviors carry over into the sync-contagion model, most notably in whether realignment will be permanent or temporary.

Furthermore, the choice of σ has opposite effects in coherent versus incoherent

systems, with regards to realignment strength. In incoherent systems, smaller values of σ allow for a tighter distribution around the target orientation $\theta_{\odot}$ during the infection period, and increase the strength of the realignment. By contrast, the tighter distribution introduced by lowering σ in coherent systems prevents susceptible individuals from accepting early infections, due to the acceptance function $\alpha(\theta)$, resulting in too many immediate transitions from susceptible to removed before sufficient realignment can occur.

We see a similar effect in varying the coupling strength κ for coherent systems. Larger values of κ lead to a tighter distribution of susceptible individuals' orientations, preventing early infections from being accepted. However, κ also controls the speed at which realignment occurs, meaning large enough κ can potentially overcome this obstacle. In incoherent systems, κ has no notable effect.

We also observed the effects of the contagion model parameters on the sync model. The infected population drives realignment of the system, so whichever choices of parameters lead to a large, sustainable infection are preferred for realignment. It is straightforward that low recovery rate γ and high initial infection probability p_0^I are always beneficial to realignment strength, due to the direct effect these have on the size of the infected population. For the remaining parameter, transmission coefficient β , the effect also depends on incoherent versus coherent behavior. For an incoherent system, a faster transmission coefficient is generally preferred, since the susceptible population will usually have individuals in range of $\theta_{\odot}$ to accept the infection. Spreading the infection faster will then create a stronger temporary realignment.

In coherent systems, the effect of β again relates to the action of the acceptance function. Early infections are typically not accepted, since the initial susceptible distribu-

tion is far from $\theta_{\odot}$. Increasing the transmission coefficient too far will then cause many of the early exposures to lead to immediate removal. The transmission coefficient must therefore be large enough to sustain the infection, countering the recovery rate, but small enough so that the system can realign closer to $\theta_{\odot}$ for infections to be accepted.

Finally, we compare our findings to those of a similar model presented by Chicoli and Paley [7]. While the two models differ in a number of ways, with ours introducing several generalizations that make the two difficult to compare, a key result of the Chicoli and Paley model is that the uniform distribution of an incoherent system can be beneficial to realignment due to a larger number of individuals already near the target orientation. We come to a similar conclusion, where the widespread distribution in an incoherent system is more reliable in propagating an infection and achieving a strong temporary realignment.

One key distinction, however, is that the Chicoli and Paley model only represents a coherent versus incoherent system through the initial conditions, and chooses a coupling strength that causes coherent dynamics in both cases. By relaxing this condition, so that the system returns to its natural coherent or incoherent state once the infection ends, we find that coherent systems have the potential to significantly outperform incoherent systems with regards to realignment success. In particular, incoherent systems can only achieve temporary realignment, since they return to an incoherent state once the infection passes. Within the context of swarm escape from a predator, though, temporary realignment towards the target orientation is likely sufficient.

One key assumption that we wish to relax in the future is the fully-mixed condition. Instead, we can introduce a network structure, where information of the infection and

orientations can only be propagated between connected neighbors. Alternatively, we can introduce planar spatial dynamics in a Vicsek-like model [46]. In this case, the network structure is dynamically changing, as individuals move and can only ‘see’ other individuals within a certain proximity. Combining a Vicsek-like model with a Kuramoto-like model has shown to result in clustering of individuals based on orientation [47, 48].

Chapter 4: A reservoir computing approach to forecasting COVID-19 data

4.1 Introduction

The COVID-19 pandemic has undoubtedly become the largest world health emergency of the modern age. Since its onset, researchers around the world have focused on developing better methods for simulation and prediction of COVID-19 data. The most successful methods have proven to be model predictors, based on the traditional SIR epidemic model [64], but augmented to consider numerous additional subgroups [69, 70] or given a network structure to better represent social interaction [71].

Predicting future values of COVID-19 case numbers falls into the larger task of time series forecasting. While modeling methods are suitable for specific forecasting tasks, general model-free machine learning methods have become increasingly popular. Strategies have involved using support vector machines [72], deep neural networks [73, 74], recurrent neural networks [75], and auto-regressive moving average models [76–78]. For the context of COVID-19 case forecasting, techniques have ranged from simple regressive models [79, 80] to more sophisticated methods, such as auto-regressive integrated moving average models [81], modified autoencoders [82], and long short-term memory recurrent

neural networks [83, 84].

Our focus is on the method of reservoir computing [85], which has gain popularity in recent years for the task of time series forecasting. Reservoir computers (RCs) have shown to accurately forecast chaotic systems [86, 87] and have the ability to learn system features such as Lyapunov exponents [88] and other statistical properties, even in non-stationary systems [89], when trained solely on data. ‘Hybrid’ reservoir computing architectures, augmented with an accompanying knowledge-based model, have been shown to vastly outperform either the RC and knowledge-based model used in isolation [90].

When applying RCs to time series forecasting with real data, two main issues arise. First, the data are unreliable in accurately representing the true state of the system. Reporting frequency leads to weekly oscillations in the reported COVID-19 case data. In most extreme cases, where reporting occurs once every few days, the data indicate several days of no cases followed by a sudden spike on a single day, immediately followed once again by no cases. We introduce two transformations to the data to combat this irregularity and increase predictability.

The second issue in applying machine learning to data-based time series prediction is a lack of quantity of data. Machine learning success typically depends on having access to a larger amount of training data. We perform our analysis using a full year of case data, but even 365 data points is small compared to typical machine learning algorithms. In addition, we should not assume that all these data are available at a time, since such an assumption would be impractical in forecasting the early stages of a future pandemic, where only a few days or weeks of data are available. We attempt to resolve this issue by considering the library-augmented reservoir computing (LARC) [91] approach, in which

the training data is supplemented by a ‘library’ of similar example time series predictors. Finally, we discuss other approaches to supplement the limited amount of available data, such as by considering past pandemics, increasing the number of input variables, or pairing the data-based RC with a model approach.

4.2 Methods

4.2.1 Data preprocessing

Successful prediction through a data-based machine learning approach requires reliable data to train the system. We address several issues present in preprocessing of the COVID-19 data set in Appendix B, and focus on our resolution within this section.

We retrieved our data from the Delphi COVIDcast Epidata API [92], which provides several U.S. COVID-19 statistics over time, including case reports, deaths, hospitalization, and bar visits, among others. While the COVIDcast data set provides several useful statistics, we focus solely on the daily number of new cases. The data are presented at different levels of geographical resolution, as small as county-level in most regions, but we will focus on state-level data and the United States as a whole.

Preprocessing the data comes in two steps: cleaning the data and transforming the data. To clean the data, we remove any invalid values. Specifically, any day with a negative number of cases reported, we set to zero cases instead. (For the prediction results we present, none of the days involve negative case counts.) For the second step of prepro-

cessing, we aim to use a transformation of the form

$$u(t) = \frac{\hat{x}'(t)}{\hat{x}(t)}, \quad (4.1)$$

where $\hat{x}(t)$ and $\hat{x}'(t)$ are estimates of the underlying continuous signal for the number of new cases $x(t)$ on day t and its derivative, respectively. The transformation Eq. (4.1) is suitable to detrend the data, removing the non-stationarity and generating a new, approximately constant, data series. Note that in the case of purely exponential growth, Eq. (4.1) would give the exponential growth rate of $x(t)$. The connection between exponential growth and epidemic spread further indicates that Eq. (4.1) would be a natural fit. However, Eq. (4.1) presents the possibility of division by zero, so we additionally increase all daily case numbers by one before applying the transformation, so that $\tilde{x}(t) = x(t) + 1$.

4.2.1.1 Windowed averaging

For our first transformation, we directly smooth to the data through an m -day windowed average. We define

$$x_{\text{avg}}(t) = \frac{1}{m} (\tilde{x}(t) + \tilde{x}(t-1) + \cdots + \tilde{x}(t-m+1)). \quad (4.2)$$

We then take $\hat{x}_{\text{avg}}(t) = x_{\text{avg}}(t-1)$ and $\hat{x}'_{\text{avg}}(t) = x_{\text{avg}}(t) - x_{\text{avg}}(t-1)$, giving a final transformation of

$$u_{\text{avg}}(t) = \frac{x_{\text{avg}}(t) - x_{\text{avg}}(t-1)}{x_{\text{avg}}(t-1)} = \frac{\frac{1}{m}(\tilde{x}(t) - \tilde{x}(t-m))}{x_{\text{avg}}(t-1)}, \quad (4.3)$$

which is equivalent to the daily percent change in the m -day average. The inverse process, to calculate $x(t)$ given $u_{\text{avg}}(t)$ and $\{x(t-1), \dots, x(t-m+1)\}$ is straightforward.

4.2.1.2 Local linear fit

As our second transformation method, we estimate $\hat{x}(t)$ and $\hat{x}'(t)$ through linear fits over an $(m+1)$ -day window. Specifically, we take $\hat{x}'(t)$ as the slope of the line fit to $\{\tilde{x}(t), \tilde{x}(t-1), \dots, \tilde{x}(t-m)\}$, given by

$$\hat{x}'_{\text{loc}}(t) = \frac{(m+1) \sum_{k=-m}^0 k \tilde{x}(t+k) - \left(\sum_{k=-m}^0 k \right) \left(\sum_{k=-m}^0 \tilde{x}(t+k) \right)}{(m+1) \sum_{k=-m}^0 k^2 - \left(\sum_{k=-m}^0 k \right)^2}. \quad (4.4)$$

For the value of $\hat{x}(t)$, we apply a linear fit to $\{\tilde{x}(t-1), \dots, \tilde{x}(t-m)\}$ and use the value of the resulting line at time $t-1$, so that

$$\hat{x}_{\text{loc}}(t) = \frac{\left(\sum_{k=-(m-1)}^0 \tilde{x}(t+k-1) \right) \left(\sum_{k=-(m-1)}^0 k^2 \right) - \left(\sum_{k=-(m-1)}^0 k \right) \left(\sum_{k=-(m-1)}^0 k \tilde{x}(t+k-1) \right)}{m \sum_{k=-(m-1)}^0 k^2 - \left(\sum_{k=-(m-1)}^0 k \right)^2}, \quad (4.5)$$

for $m > 1$. For $m = 1$, we take $\hat{x}_{\text{loc}}(t) = \tilde{x}(t-1)$. Together, we have the transformation

$$u_{\text{loc}}(t) = \frac{\hat{x}'_{\text{loc}}(t)}{\hat{x}_{\text{loc}}(t)}. \quad (4.6)$$

The inverse transformation to recover $x(t)$ is again straightforward. We have intentionally chosen to use one less data point in determination of the denominator ($\hat{x}_{\text{loc}}(t)$) to guarantee invertability.

Note that we have defined m such that for $m = 1$, both transformations give the daily percent change

$$u(t) = \frac{\tilde{x}(t) - \tilde{x}(t-1)}{\tilde{x}(t-1)}. \quad (4.7)$$

We present examples of both transformations in Figure 4.1 for the national-level data and Figure 4.2 for state-level data from New York and Kansas. As m increases, both transformations reduce the magnitude of the oscillations by approximately the same amount. Of special note is the large spikes present in the Kansas data, which are removed by increasing the smoothing parameter m . The irregular reporting in the Kansas data that created these spikes would otherwise present an issue in its predictability.

4.2.2 Reservoir computer architecture

Reservoir computing is a machine learning approach involving a large recurrent network (a ‘reservoir’) that learns a mapping of a much smaller-scale system [85]. A main advantage of reservoir computing, compared with other machine learning techniques, is the simplicity in the training process. Specifically, the majority of the reservoir computer system parameters are set randomly, based on predetermined distributions, with only the map from the reservoir to the output value being changed through the training process. The details in the implementation of the reservoir computer can vary, particularly in the initialization of different features, so we focus only on our implementation.

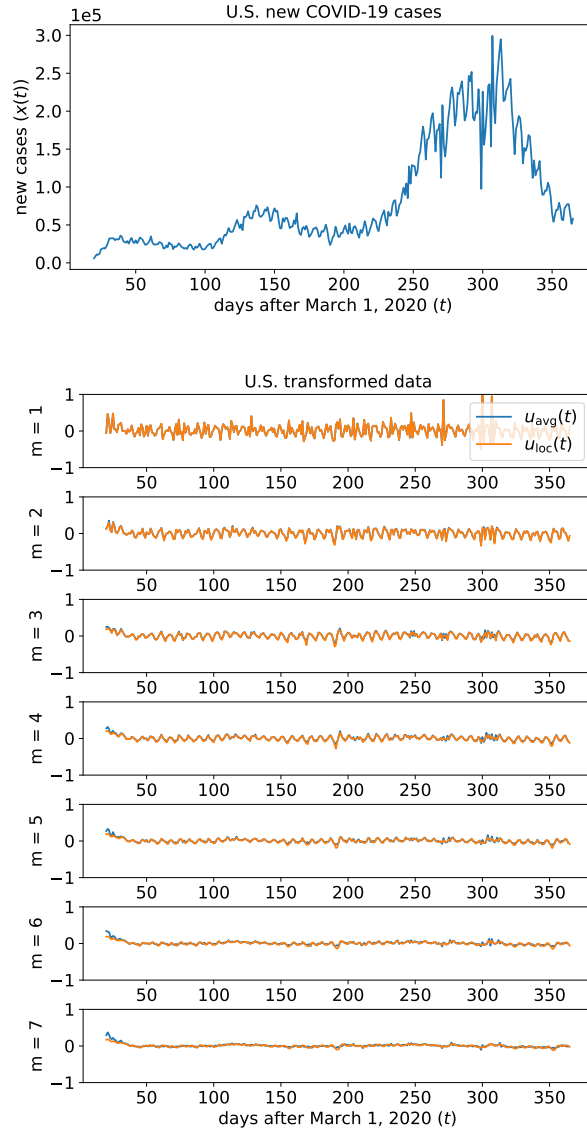


Figure 4.1: **Top:** New COVID-19 cases $x(t)$ for the United States t days after March 1, 2020 shown from March 21, 2020 ($t = 20$) to February 28, 2021 ($t = 364$). **Bottom:** Transformations of the U.S. case data using the windowed average (blue) and the local linear fit (orange) methods for different choices of m . For $m = 1$, the two transformations are identical, but even for larger m the transformations give similar results. Note that larger m leads to greater ‘smoothing’ of the data, causing increased suppression of the oscillations.

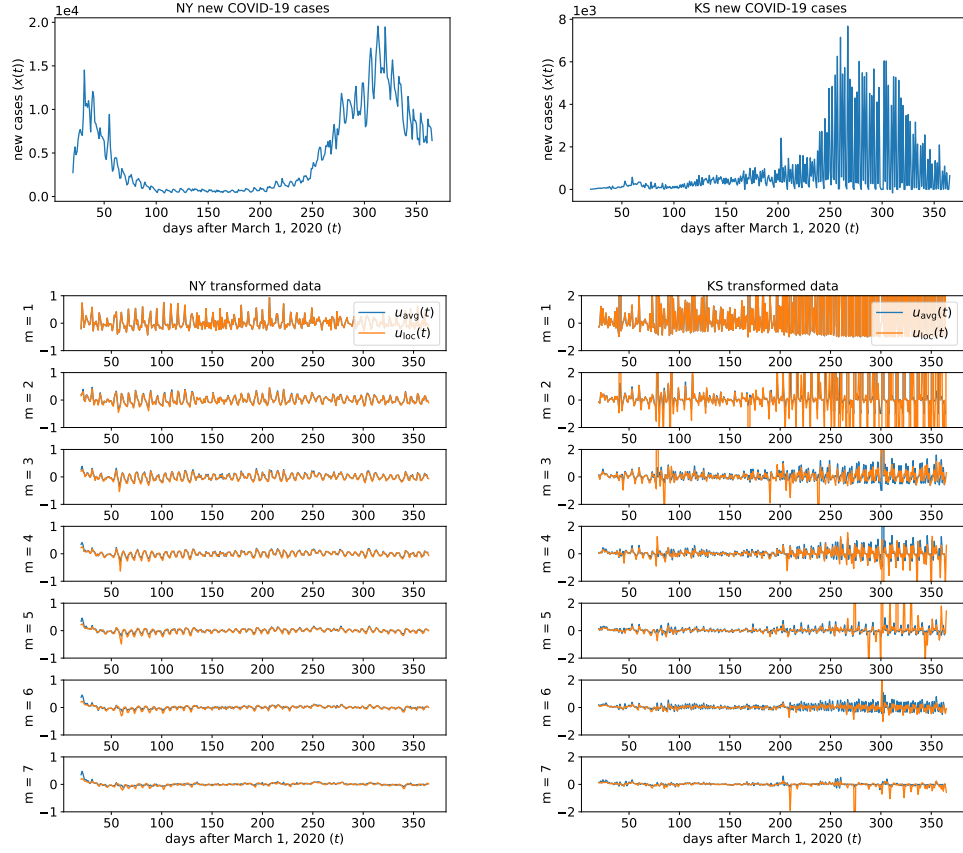


Figure 4.2: **Top:** New COVID-19 cases $x(t)$ for New York (left) and Kansas (right) t days after March 1, 2020 shown from March 21, 2020 ($t = 20$) to February 28, 2021 ($t = 364$). **Bottom:** Transformations of the New York and Kansas case data using the windowed average (blue) and the local linear fit (orange) methods for different choices of m . Note that the two transformations are identical for $m = 1$. For larger values of m , the transformations give similar results for well-behaved data, like New York and the United States data (Figure 4.1). For poor-quality data, such as for Kansas, the two transformations are more distinct. The windowed average approach can more effectively remove massive spikes, but is less effective at overall suppression of the resulting oscillations (see for example $m = 6$).

A reservoir computer consists of three layers: the input layer, the reservoir, and the output layer. The reservoir consists of a directed network of D nodes, where D is large, with adjacency matrix $\mathbf{A}$. In our implementation, we generate the reservoir as a sparse Erdős-Rényi network with mean degree d . The non-zero values of the adjacency matrix are initially randomly drawn from a standard normal distribution, but then all values of $\mathbf{A}$ are uniformly scaled so that the spectral radius is ρ . The state of the reservoir can be represented by the D -dimensional vector $\mathbf{r}(t)$, where $r_i(t)$ is the scalar state of node i at time t .

An N -dimensional input vector $\mathbf{u}(t)$ is coupled to the reservoir through a $D \times N$ matrix $\mathbf{W}_{\text{in}}$ and the update equation

$$\mathbf{r}(t+1) = \tanh [\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{\text{in}}\mathbf{u}(t) + \mathbf{b}], \quad (4.8)$$

where $\tanh[\cdot]$ is taken element-wise and $\mathbf{b}$ is a D -dimensional *bias vector*. (In the case of one-dimensional input, we will use $\mathbf{u}(t)$ and $u(t)$ interchangeably.) The weights of $\mathbf{W}_{\text{in}}$ are randomly generated from a uniform distribution over $[-\sigma, \sigma]$, and the weights of $\mathbf{b}$ are taken from a uniform distribution over $[-\beta, \beta]$.

The reservoir is then mapped to an output vector, which in the case of time series forecasting is the input vector at the next time step $\mathbf{u}(t+1)$. The map from the reservoir to the output is given by

$$\mathbf{u}(t+1) = \mathbf{W}_{\text{out}}\mathbf{r}(t+1), \quad (4.9)$$

for some $N \times D$ matrix $\mathbf{W}_{\text{out}}$. After initialization, only $\mathbf{W}_{\text{out}}$ is altered, which occurs

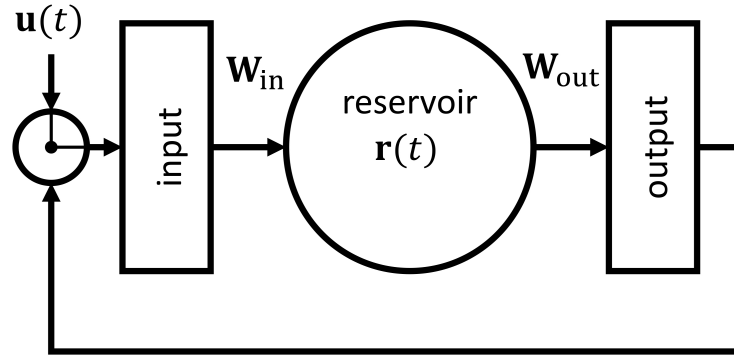


Figure 4.3: The structure of a standard reservoir computer. During ‘open loop’ training, the switch on the left takes in the input $u(t)$. The input is coupled to the reservoir through the input matrix W_{in} and update Eq. (4.9). The output is then generated by multiplying the reservoir state $r(t + 1)$ by the output matrix W_{out} . After training, the switch on the left is rotated to accept the output of the reservoir computer as the input for the next time step, creating an autonomous ‘closed loop’.

during the training phase.

In the training phase, data are fed into the reservoir computer while it is kept in ‘open loop’ mode, in which the reservoir states are tracked, but the outputs are not looped back into the reservoir as input. After throwing away an initial transient, the reservoir states are used to fit W_{out} so that the output mapping Eq. (4.9) best approximates the true values of the training data, typically determined through a least squares fit. Once trained, the reservoir can then be switched to ‘closed loop’ prediction mode, in which the outputs of the reservoir are used as the subsequent inputs, creating a fully autonomous system. A full schematic of a reservoir computer is presented in Figure 4.3.

4.2.3 Recursive least squares

To fit $\mathbf{W}_{\text{out}}$ to the training data, we will employ the recursive least squares method.

In the recursive least squares method, $\mathbf{W}_{\text{out}}$ is chosen to minimize the cost function

$$\min_{\mathbf{W}_{\text{out}}} \left\{ \sum_{j=0}^{k-1} \lambda^{k-j} \|\mathbf{W}_{\text{out}} \mathbf{r}(t_j) - \mathbf{u}(t_j)\| \right\}, \quad (4.10)$$

where $t_0 < t_2 < \dots < t_{k-1}$ and $0 < \lambda \leq 1$ is the ‘forgetting factor’. Traditionally, recursive least squares is used for online training, where $\mathbf{W}_{\text{out}}$ is continuously updated after each time step as new data come in. For our purposes, though, we are interested in the exponential discounting by λ .

For non-stationary data, such as the COVID-19 case data, the underlying hidden dynamics are constantly changing. We will assume, however, that the data are locally stationary, in which any underlying parameters are approximately constant and short-term predictions can accurately be based on recent data, but not necessarily data far into the past. We do not want to completely ignore the older data, especially since we are limited to very few available data points. The solution then is to exponentially-weight the data, as in the recursive least squares method, so that the most recent data points are more heavily considered in determining $\mathbf{W}_{\text{out}}$ without totally losing old values.

4.2.4 Baseline predictors

For comparison to the reservoir computing approach, we will use a polynomial predictor applied directly to the daily case data $x(t)$. For small windows, the mean behavior

of the daily case data is approximately linear, suggesting a straight-line fit may be sufficient for short-term predictions of the mean behavior. We also consider a quadratic fit as a slightly higher-order alternative.

4.2.5 Library-augmented reservoir computing

As an attempt to augment the limited data set, we will also consider the library-augmented reservoir computing (LARC) process [91]. As a ‘metalearning’ approach, LARC involves using information from a ‘library’ of trained reservoir computers to construct a new reservoir computer that will predict a time series signal not yet seen by the library.

The library consists of identical reservoir computers, with the same reservoir topology and randomized system variables. However, each library member is trained on different training data, resulting in a different sequence of reservoir states $\mathbf{r}(t)$ and a different output mappings $\mathbf{W}_{\text{out}}$. Let $\mathbf{r}_0$ denote the state of the reservoir at the end of training. The set of pairs $\{\mathbf{r}_0^i, \mathbf{W}_{\text{out}}^i\}$, for each library member i , is used to train an autoencoder, which embeds these high-dimensional reservoir state and output matrix pairs in a low-dimensional latent space. The latent space embedding can then be ‘decoded’ to produce a close approximation of the original input.

When presented with new time series data, instead of directly training a reservoir computer, we can search the latent space for a state $\mathbf{e}$, which when decoded, gives the pair $(\tilde{\mathbf{W}}_{\text{out}}, \tilde{\mathbf{r}}_0)$ that produce a reservoir computer that best predicts the given time series. Note that this reservoir computer is otherwise identical to those used by library members. All

else equal, the defining characteristics of a reservoir computer is the initial state $\mathbf{r}_0$ before prediction and the output mapping $\mathbf{W}_{\text{out}}$. A schematic of the full LARC process is shown in Figure 4.4.

This library approach is particularly useful when a new time series is too short to reliably train a reservoir computer directly, such as at the onset of a pandemic, where available data is limited, but where one may have access to past data with similar dynamics that can be used to construct a library (e.g. a past pandemic).

4.2.6 Predictor evaluation

We consider two measures of success for our different predictors. The first measure, the cumulative error, gives the mean prediction accuracy, while the second measure, the normalized mean absolute error, tracks errors within individual daily predictions.

4.2.6.1 Cumulative error

In the context of epidemic response, using the cumulative case count is a reasonable choice, since we primarily care about the total number of new cases that appear over a short period of time, rather than the specific number of cases that occur on each day. Therefore, we first consider the percent cumulative error,

$$\text{err}_{\text{cumul}} = \frac{|\sum_{t=t_0}^{t_0+T-1} x_{\text{truth}}(t) - \sum_{t=t_0}^{t_0+T-1} x_{\text{pred}}(t)|}{\sum_{t=t_0}^{t_0+T-1} x_{\text{truth}}(t)}, \quad (4.11)$$

where t_0 is the starting day of the prediction, T is the prediction length, $x_{\text{truth}}(t)$ is the true number of cases on day t , and $x_{\text{pred}}(t)$ is the predicted number of cases on day t . We will

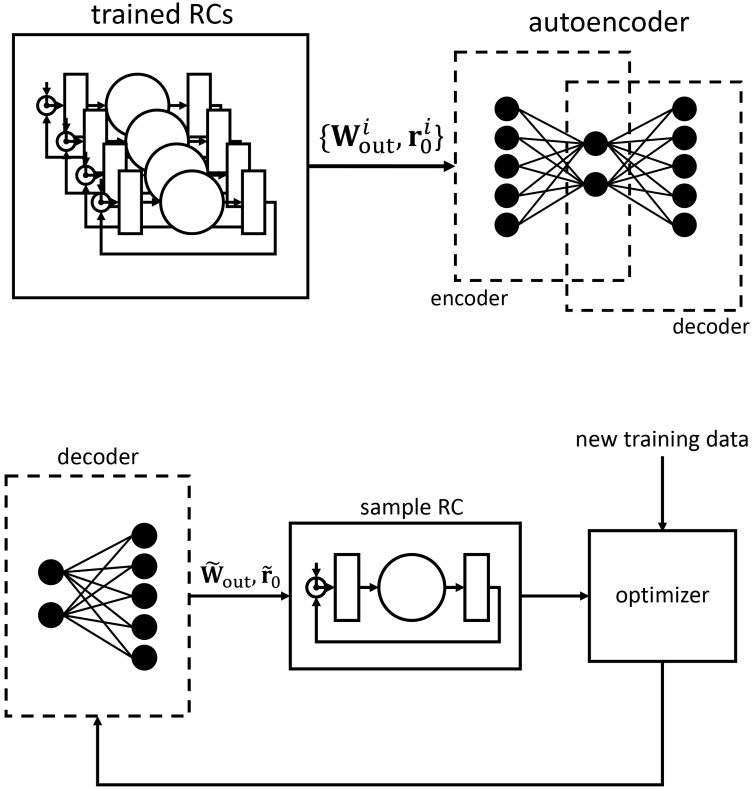


Figure 4.4: The two stages of the library-augmented reservoir computing process. In stage one (top), a library of trained reservoir computers are used to train an autoencoder to predict the pair $(\mathbf{W}_{\text{out}}, \mathbf{r}_0)$, consisting of an output mapping $\mathbf{W}_{\text{out}}$ generated from training the RC and an associated reservoir state $\mathbf{r}_0$ at the end of training. In stage two (bottom), a state from the embedded space of the autoencoder is decoded to generate a pair $(\tilde{\mathbf{W}}_{\text{out}}, \tilde{\mathbf{r}}_0)$. This pair is used to construct a new reservoir computer, that is otherwise identical to those in the library. The sample reservoir computer generates a signal that is compared to a new (short) signal that we wish to predict. Using an optimization scheme, information is sent back to the decoder to pick a new pair $(\tilde{\mathbf{W}}_{\text{out}}, \tilde{\mathbf{r}}_0)$ that better matches the new data until some stopping criterion is reached.

primarily focus on the cumulative error after one week $T = 7$, to match the perceived one-week period of oscillation in the daily case numbers.

Using the cumulative case count allows for small errors in the specific daily predictions to even out, particularly when they are alternating over/under-estimates from the weekly oscillations being off by one day. Furthermore, using cumulative cases allows us to compare to a linear predictor, which will be unable to closely capture the oscillations in the data, but can potentially match the mean behavior.

4.2.6.2 Normalized mean absolute error

Although cumulative counts are preferable, we still desire a measure of the daily prediction accuracy. To this end, we use the mean absolute error between the true daily cases $x_{\text{truth}}(t)$ and the predicted daily cases $x_{\text{pred}}(t)$, normalized by the mean of the true data measured over the same period,

$$\text{NMAE} = \frac{\frac{1}{T} \sum_{t=t_0}^{t_0+T-1} |x_{\text{truth}}(t) - x_{\text{pred}}(t)|}{\frac{1}{T} \sum_{t=t_0}^{t_0+T-1} x_{\text{truth}}(t)}. \quad (4.12)$$

We again choose to use a one week window ($T = 7$) to match the one-week period in the daily case oscillation.

4.3 Results

In the following section, we present our main results. Unless otherwise specified, all reservoir computers use the hyperparameters listed in Table 4.1. Note that the ‘initialization length’ refers the number of points $\mathbf{u}(t)$ prior to the training data that are used

Hyperparameter	Value
reservoir size (D)	500
mean degree (d)	3
spectral radius (ρ)	0.7
input scaling distribution hyperparameter (σ)	5.75
bias distribution hyperparameter (β)	0.1
forgetting factor (λ)	0.96
initialization length	10
training length	21

Table 4.1: List of hyperparameter values used in training the reservoir computers, unless otherwise noted.

to synchronize the reservoir state, but which are not used for fitting $\mathbf{W}_{\text{out}}$. The ‘training length’ refers to the number of data points $\mathbf{u}(t)$ in the training data used to fit $\mathbf{W}_{\text{out}}$. It should be noted that the hyperparameter choices were optimized ‘by hand’, by manually adjusting each hyperparameter until no noticeable improvement could be gained. The results can potentially be further improved by an automated optimization scheme for the hyperparameters.

4.3.1 Recursive least squares and forgetting factor selection

We begin by determining an appropriate forgetting factor λ to be used for reservoir computer training. For each starting value of $t_0 \in [50, 150]$, we construct a reservoir computer trained on the entire history $\{u(0), u(1), \dots, u(t_0 - 1)\}$, with the first ten points used to initialize the reservoir, resulting in 101 sample predictions. Data are transformed using the daily percent change Eq. (4.7) to calculate each $u(t)$. We repeat this task for different values of λ . Smaller values of λ correspond to faster forgetting rates, meaning older data is less influential for smaller λ , at the cost of requiring less accuracy overall in the fit of $\mathbf{W}_{\text{out}}$.

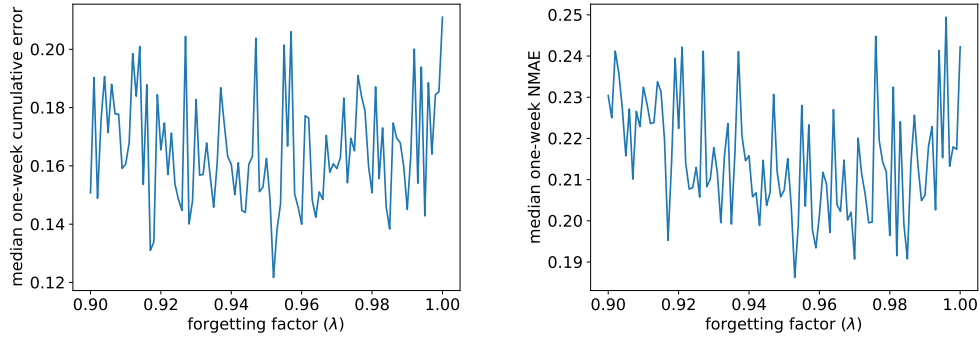


Figure 4.5: Reservoir computer predictor results for different choices of forgetting factor λ . For each value of λ shown, we increment through all prediction start times t_0 in $[50, 150]$. For each value of t_0 , a reservoir computer was trained on the entire history (i.e., all data from $t = 0$ to $t = t_0 - 1$), with data transformed according to the percent daily change Eq. (4.7).

In Figure 4.5, we present the median values of the one-week cumulative error and one-week NMAE, as describe in Section 4.2.6, for $0.9 \leq \lambda \leq 1$ in steps of 0.001. Note that $\lambda = 1$ corresponds to no forgetting, i.e., a standard linear regression. While the values of our measures fluctuate wildly, there is a noticeable dip in the NMAE near $\lambda = 0.96$, indicating that a small amount of forgetting is potentially beneficial.

4.3.2 Predictor results

We now present the main results of our four prediction methods. To begin, we consider sample predictions at specific points in time using the U.S. case data, shown in Figure 4.6, and all fifty states plus the District of Columbia, shown in Figures 4.7 and 4.8. The main distinction with the reservoir computer approaches is the ability to learn the oscillations in the data, leading to perceived better accuracy in the daily predictions. Noticeably, the amplitude of oscillations in the predictions for the RC trained on the windowed average transformed data tends to grow in magnitude over time. Since we are

mainly interested in short-term predictions (up to seven days), the growing oscillations are not a major concern. All four methods tend to follow the general mean behavior of the data.

Even when considering the state data (Figures 4.7 and 4.8), we again see that the reservoir computer predictor is able to match the oscillations in the data quite effectively, which is an impossible task for a linear extrapolation. We do note that several states have failed reservoir computer predictions that ‘blow up’ almost immediately. It is not immediately clear what the cause of this blow-up is, but one possible explanation is a poor choice of RC hyperparameter values (Table 4.1). Ideally, the hyperparameters would be individually selected for each state to best match the data; however, doing so by hand for every state would be time consuming. The hyperparameter choices we use work well for the majority of the states, but the results would be aided by automated hyperparameter optimization.

To validate these initial observations, we turn to the distributions of error presented in Figure 4.9 for predictions generated on the U.S. data set, Figure 4.10 for the New York data set, and Figure 4.11 for the Kansas data set. We generate these results by producing a prediction starting at each time $t_0 \in [50, 250]$, for a total of 201 sample predictions. We consider New York and Kansas as prime examples of the two extremes of data quality: the New York case data are similar in quality to the United States total data, while the Kansas data consist of irregular reporting that creates additional challenges for forecasting.

As observed in the example predictions, all four methods perform fairly well in predicting the one-week cumulative case numbers, indicating good fit to the mean behavior, with the reservoir computer methods performing slightly better. The worst predictor

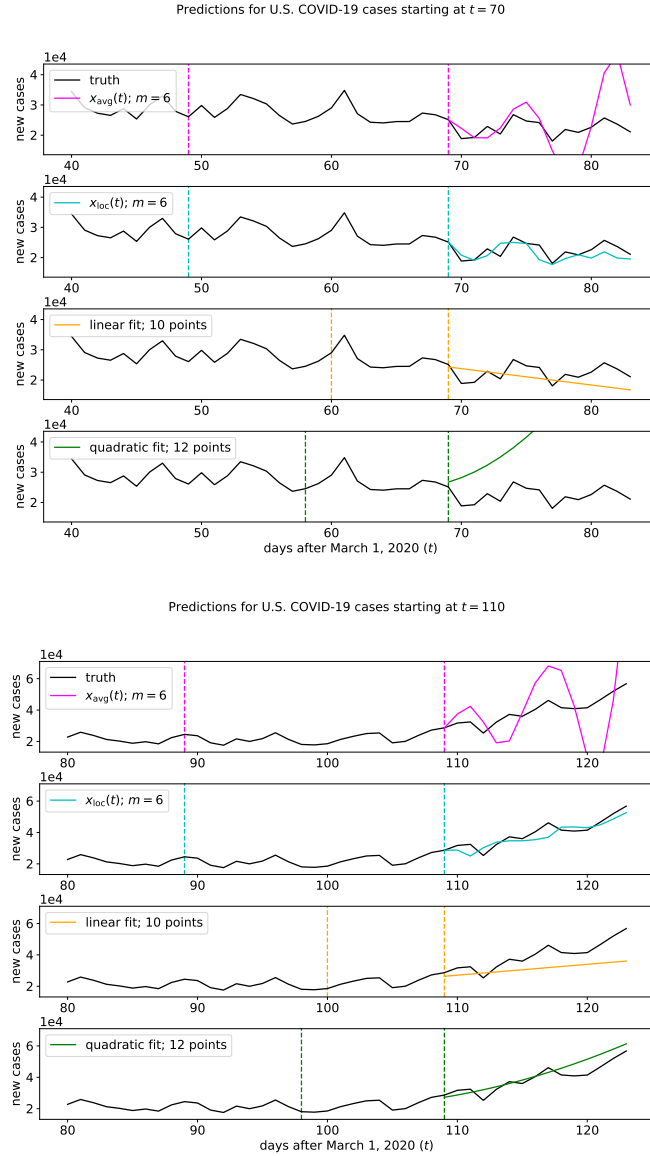


Figure 4.6: The true daily U.S. COVID-19 case data (black) and example predictions for the four prediction methods: reservoir computer with the windowed average transformation applied to daily cases with $m = 6$ (magenta), reservoir computer with the local linear fit transformation applied to daily cases with $m = 6$ (cyan), a linear fit applied to the daily case data using 10 points (orange), and a quadratic fit applied to the daily case data using 12 points. Predictions start at times $t = 70$ (top) and $t = 110$ (bottom). Vertical dashed lines mark the start and end of data points ‘visible’ to the corresponding predictors.

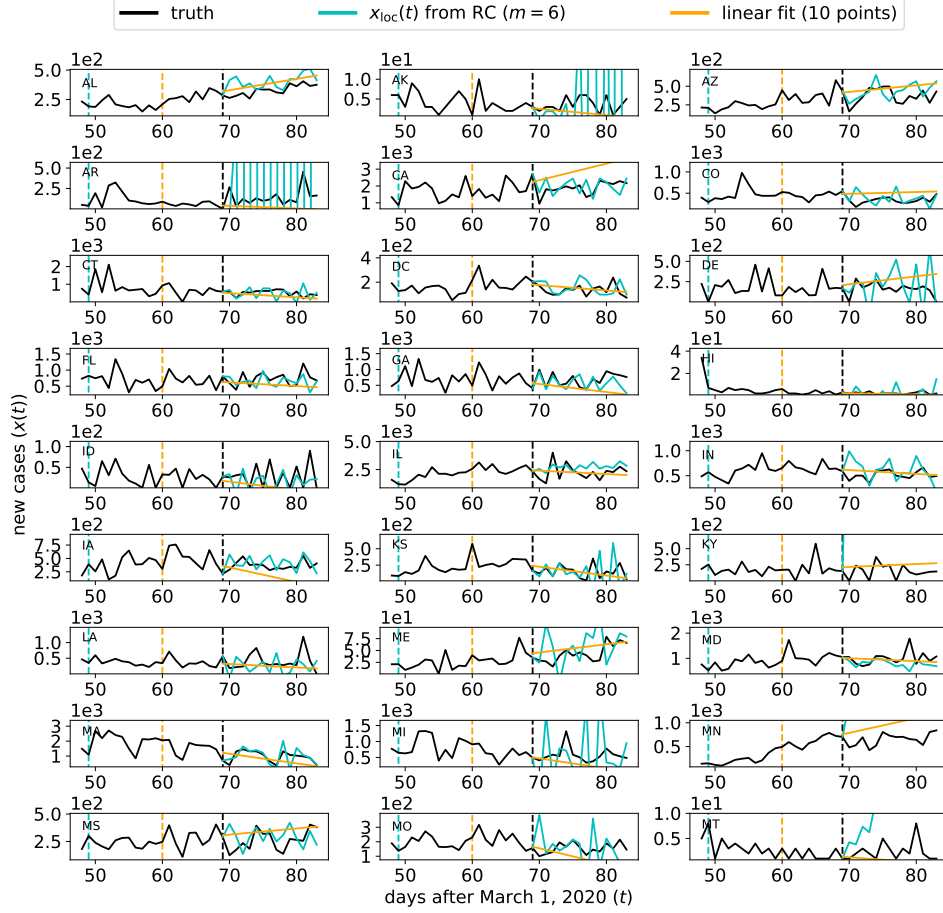


Figure 4.7: State-level COVID-19 case data (black) compared to a reservoir computer prediction using the local linear fit transformation (cyan) with $m = 6$ and a linear extrapolation (orange) using 10 points. All predictions start on day 70, the first day after the vertical dashed black line. The vertical orange dashed line marks the start of points used by the linear extrapolation and the vertical cyan dashed line marks the start of points used to train the reservoir computer. Here we show the first twenty-six states alphabetically and the District of Columbia. The remaining states are shown in Figure 4.8.

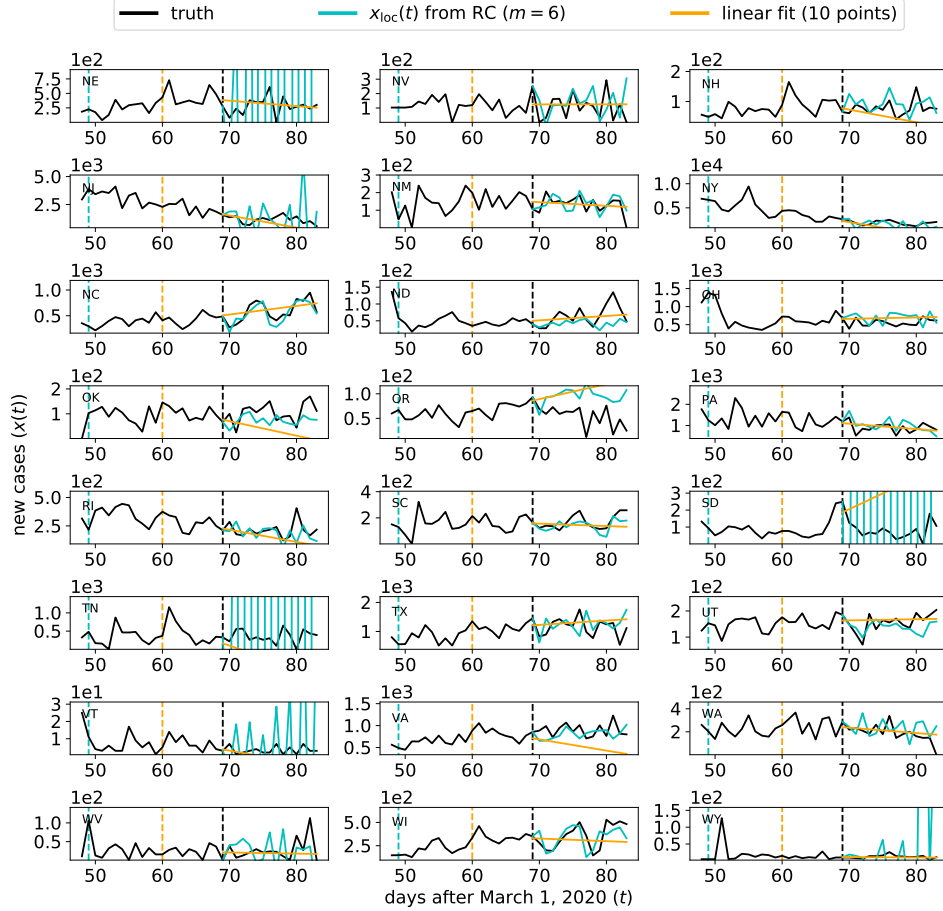


Figure 4.8: The continuation of Figure 4.7, showing state-level COVID-19 case data (black) compared to a reservoir computer prediction using the local linear fit transformation (cyan) with $m = 6$ and a linear extrapolation (orange) using 10 points. All predictions start on day 70, the first day after the vertical dashed black line. The vertical orange dashed line marks the start of points used by the linear extrapolation and the vertical cyan dashed line marks the start of points used to train the reservoir computer.

appears to be the quadratic fit, which is more likely to quickly diverge from the true data in the case of a bad fit, and diverges at a quadratic rate.

It is difficult to determine a direct comparison between the RC predictors and the polynomial fit predictors, since the values of m used in the data transformations do not correspond well to the number of points used in the polynomial fits. Furthermore, the polynomial fits are directly applied to the raw case data, while the reservoir computer predictions include an additional data transformation step. While it is true that the value of m relates to the number of points used in the smoothing of the data, it does not include the 21 days of training advantage of the reservoir computer. Despite this advantage, the reservoir computer predictions in general do not appear to be significantly better than a simple linear extrapolation. When presented with a small amount of data, it may perhaps then be reasonable to assume a linear growth rate over the short-term.

While we may be unable to directly compare between the predictors for a fixed number of points, we can compare the best-case results. From our national, New York, and Kansas results, we estimate that the best reservoir computer predictor is the local linear fit transformation with $m = 6$, and the best polynomial fit predictor is the linear fit with 10 points. In Figures 4.9(c,f), 4.10(c,f), and 4.11(c,f), we include a direct comparison between the distributions of these best performers. In addition, in Figure 4.12, we produce a scatter plot comparison of the prediction errors for these two methods when applied to each state, the District of Columbia, and the United States case data. We find that most points lie near the line $y = x$, demonstrating no significant improvement in predictive power from one method or the other. We do note that the reservoir computer prediction does perform slightly better at predicting the cumulative number of cases for almost every

Results for U.S. COVID-19 case predictions

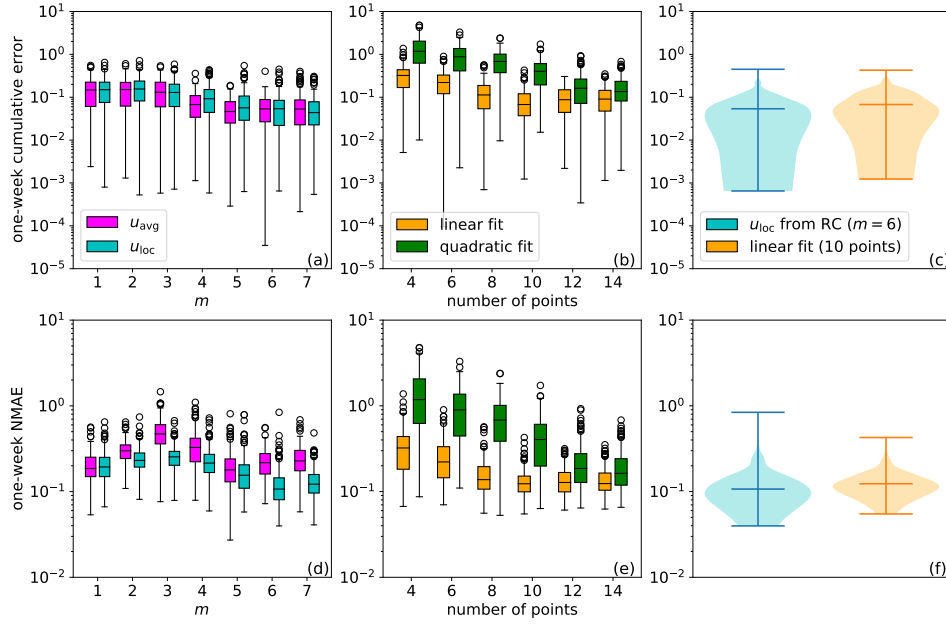


Figure 4.9: Predictor statistics for the four different predictor methods with different choices of m for the reservoir computer transformations (a,d) and number of points for the polynomial fits (b,e) applied to the U.S. COVID-19 daily case data. Statistics are taken by generating a prediction for each starting time t_0 in $[50, 250]$. In (c,f), we compare the distributions for the errors of the reservoir computer predictions using the local linear fit transformation with $m = 6$ and the linear extrapolation using 10 points.

Results for New York COVID-19 case predictions

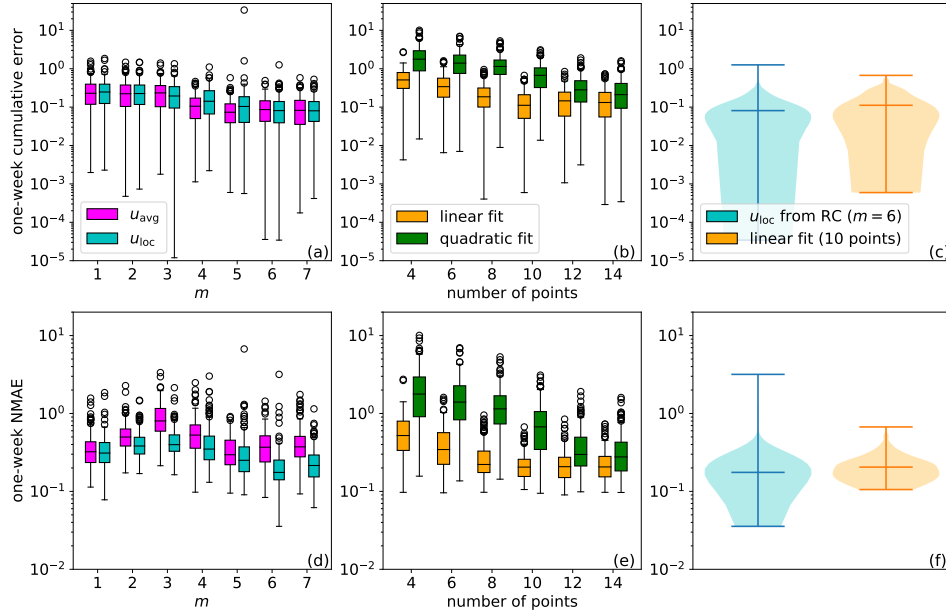


Figure 4.10: Predictor statistics for the four different predictor methods with different choices of m for the reservoir computer transformations (a,d) and number of points for the polynomial fits (b,e) applied to the New York COVID-19 daily case data. Statistics are taken by generating a prediction for each starting time t_0 in $[50, 250]$. In (c,f), we compare the distributions for the errors of the reservoir computer predictions using the local linear fit transformation with $m = 6$ and the linear extrapolation using 10 points.

Results for Kansas COVID-19 case predictions

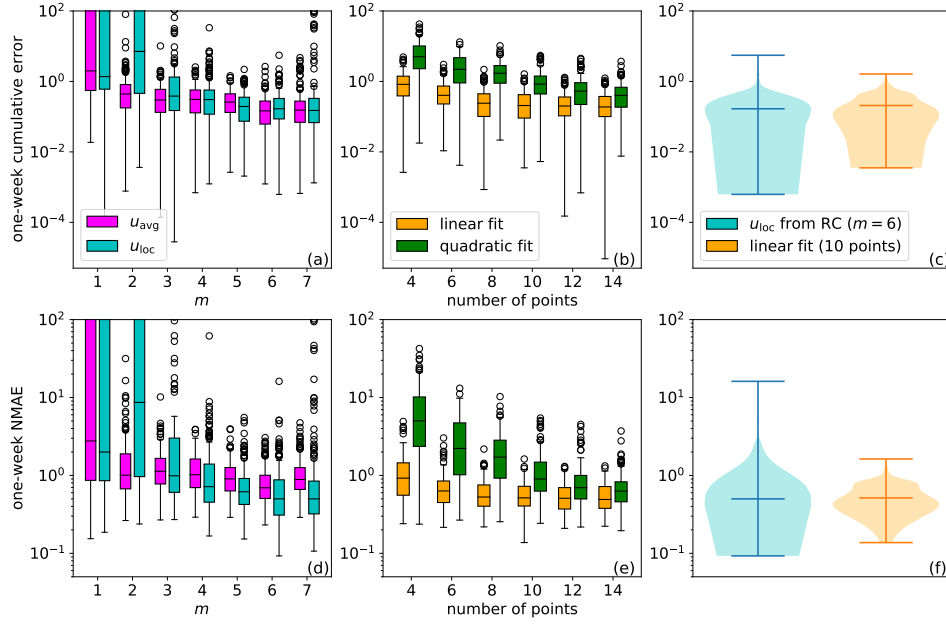


Figure 4.11: Predictor statistics for the four different predictor methods with different choices of m for the reservoir computer transformations (a,d) and number of points for the polynomial fits (b,e) applied to the Kansas COVID-19 daily case data. Statistics are taken by generating a prediction for each starting time t_0 in $[50, 250]$. In (c,f), we compare the distributions for the errors of the reservoir computer predictions using the local linear fit transformation with $m = 6$ and the linear extrapolation using 10 points. Note that the box plots for the reservoir computer predictors (a,d) with low values of m extend way beyond the vertical scale used. A zoom out is provided in Figure 4.13.

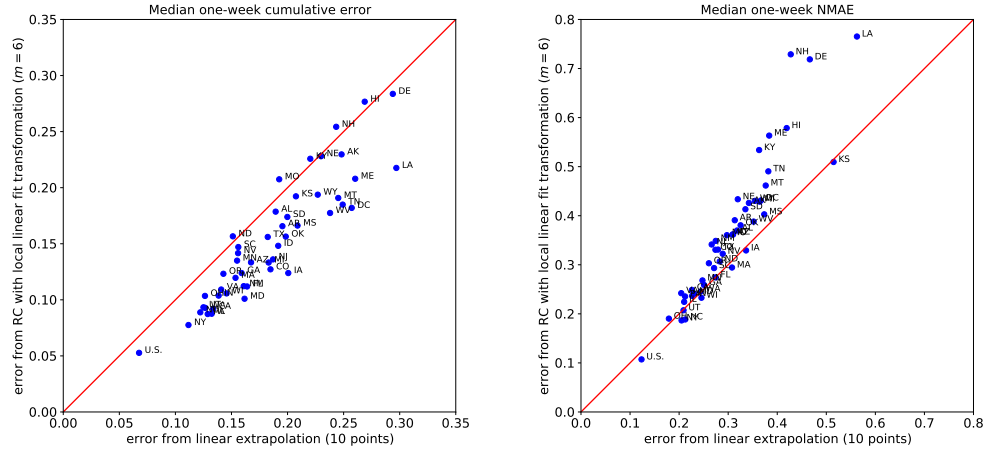


Figure 4.12: Scatter plots comparing the one-week cumulative error (left) and one-week NMAE (right) for the reservoir computer prediction using the local linear fit transformation with $m = 6$ and the linear extrapolation using 10 points. The coordinates of each point are the median values of the error metrics for the labeled locations. The red line marks $y = x$. The plot includes forty-seven states, the District of Columbia, and the United States as a whole. Not included are Connecticut, Rhode Island, and Vermont, which each had significantly higher median errors from both predictors.

state, indicating better prediction of the mean behavior. Perhaps counter-intuitively, the reservoir computer does worse at predicting the individual daily cases numbers, as measured through the NMAE, despite being able to apparently match the daily oscillations better than a linear extrapolation. A probable cause is the oscillations in the predictions being slightly ‘out-of-phase’ with the true oscillations, which increases the daily errors but has no impact on the overall cumulative count.

Deviating from the discussion of comparing predictors for a moment, we turn our attention to the results of the Kansas data set, shown in Figures 4.11 and 4.13. The Kansas data is quite poor in quality, when considered for a prediction task. The irregular reporting frequency leads to several days of zero cases reported, followed by a large spike in cases. The smoothing induced by our transformations are able to salvage the data. With a small

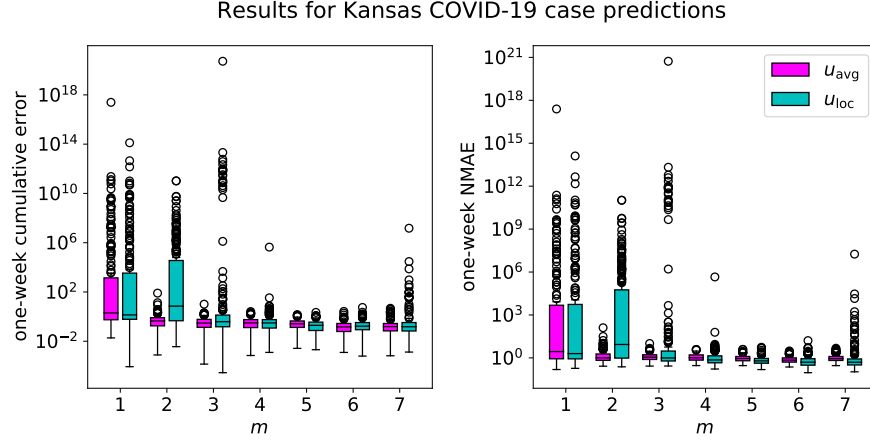


Figure 4.13: Zoom out of Figure 4.11(a,d), showing the full distributions for low values of m .

amount of smoothing ($m = 4$), the reservoir computer prediction accuracy of the Kansas data is of similar order to the well-behaved data sets of New York and the whole U.S. By returning to Figure 4.2, we can see that $m = 4$ is when the majority of spikes in the data set are no longer present over the timespan of $[50, 250]$. While proper data transformation is key in allowing for successful RC prediction, we again note that the predictions are not exceptionally better than a direct linear fit.

4.3.3 Library-augmented reservoir computing applied to U.S. COVID-19 case data

An alternative to a simple reservoir computer predictor is the metalearning approach of library-augmented reservoir computing (LARC). To apply LARC, we first must construct a library of reservoir computers that will be used to train an autoencoder. For our investigation, each library member consists of a single RC trained on a 50-day window of data, with the first 10 days of each window used for reservoir initialization. Each li-

library member uses a different window, but these windows may overlap. Data input into the reservoir computers are transformed according to the daily percent change Eq. (4.7) (equivalent to either transformation method with $m = 1$). Note that all library members use identical reservoir computers (in terms of network topology and other ‘randomized’ variables). All hyperparameters, other than the training length are identical to those specified in Table 4.1.

Before using LARC to generate predictions, we first validate the assumption that the latent embedding of the library results in clustering of similarly-behaved systems. That is, two library members close in the latent space will correspond to similar time series. We test this assumption using the U.S. data set, creating a library member for each window of data starting in $t \in [30, 300]$. We embed the library in a three-dimensional latent space, for visualization purposes. The resulting embedding is shown in Figure 4.14, where points are colored according to the training window’s start time, and demonstrates a clear clustering by this value.

For evaluation of LARC-generated predictions, we make a few modifications. First, we increase the dimension of the latent space to twelve, and second, we restrict our library members to consist of windows starting in $[30, 125]$. We then generate predictions with starting times $t_0 \geq 180$ using the LARC method with new signals containing only five data points. Note that a new signal containing any points before day 175 would otherwise have some overlap with at least one library member’s window.

In Figure 4.15, we present several sample predictions generated by LARC and compare them to a single reservoir computer prediction generated from 21 days of training data, as well as a linear extrapolation using only the five days of information. Qualita-

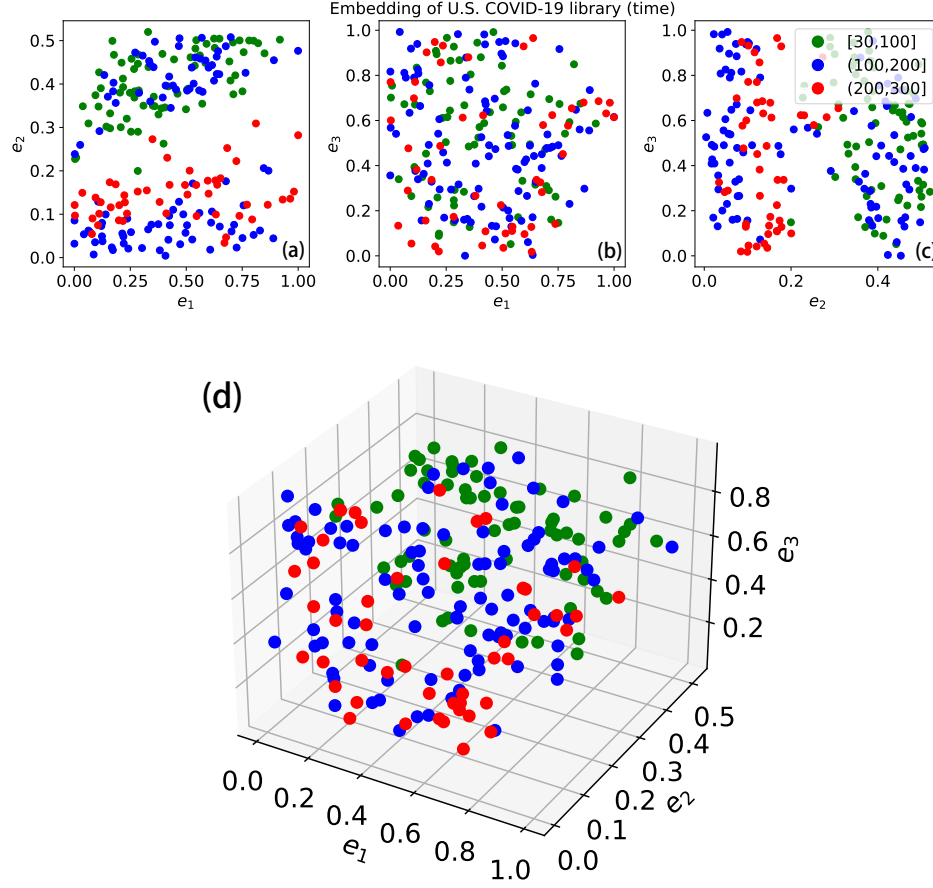


Figure 4.14: Embedding of the U.S. COVID-19 library into a three-dimensional latent space using an autoencoder. The library consists of reservoir computers trained over 50-day overlapping windows of data from $t = 30$ to $t = 300$. Daily case data were transformed using the percent daily change Eq. (4.7) before input into the RCs. Points are colored according to the first day of the training window, with green points corresponding to starting in $[30, 100]$, blue points corresponding to starting in $(100, 200]$, and red points corresponding to starting in $(200, 300]$. Panels (a–c) are projections of the points onto the three coordinate planes. Panel (d) shows all points in three-dimensions.

tively, the reservoir computer predictions generated from the five-day signals (aided by LARC) are similar to those generated from a 21-day training signal. When viewing the seven-day zoom in, the predictions between LARC applied to a five-day signal and a single RC with a 21-day signal are nearly identical in several cases. The linear extrapolation based on five points appears to have difficulty matching the mean behavior, due to seeing approximately half of the weekly oscillation. For instance, if the case numbers are mainly increasing towards the peak of the week over the five day period, then the linear extrapolation will predict a continuation of this increase, as seen in the first prediction in Figure 4.15.

The power of LARC comes from being able to replace a large training data set with a small data set supplemented by a large library. This advantage would be particularly useful in the early stages of a pandemic, where only a few days of data are available. In this case, one would not be able to generate a library out of data from the current pandemic, but could instead rely on data from past pandemics. A signal that is better-represented by the library will produce a higher-quality prediction.

4.3.4 Autoencoder embedding of state-level COVID-19 data

We can perform a similar treatment to state-level COVID-19 data as done for the U.S. data set to validate the assumption that time series with similar behavior would be close in latent space, thereby indicating LARC as a suitable forecasting approach. We construct a full library in the same way as before, taking reservoir computers trained on overlapping 50-day windows with start times in $[30, 300]$. We complete this process with

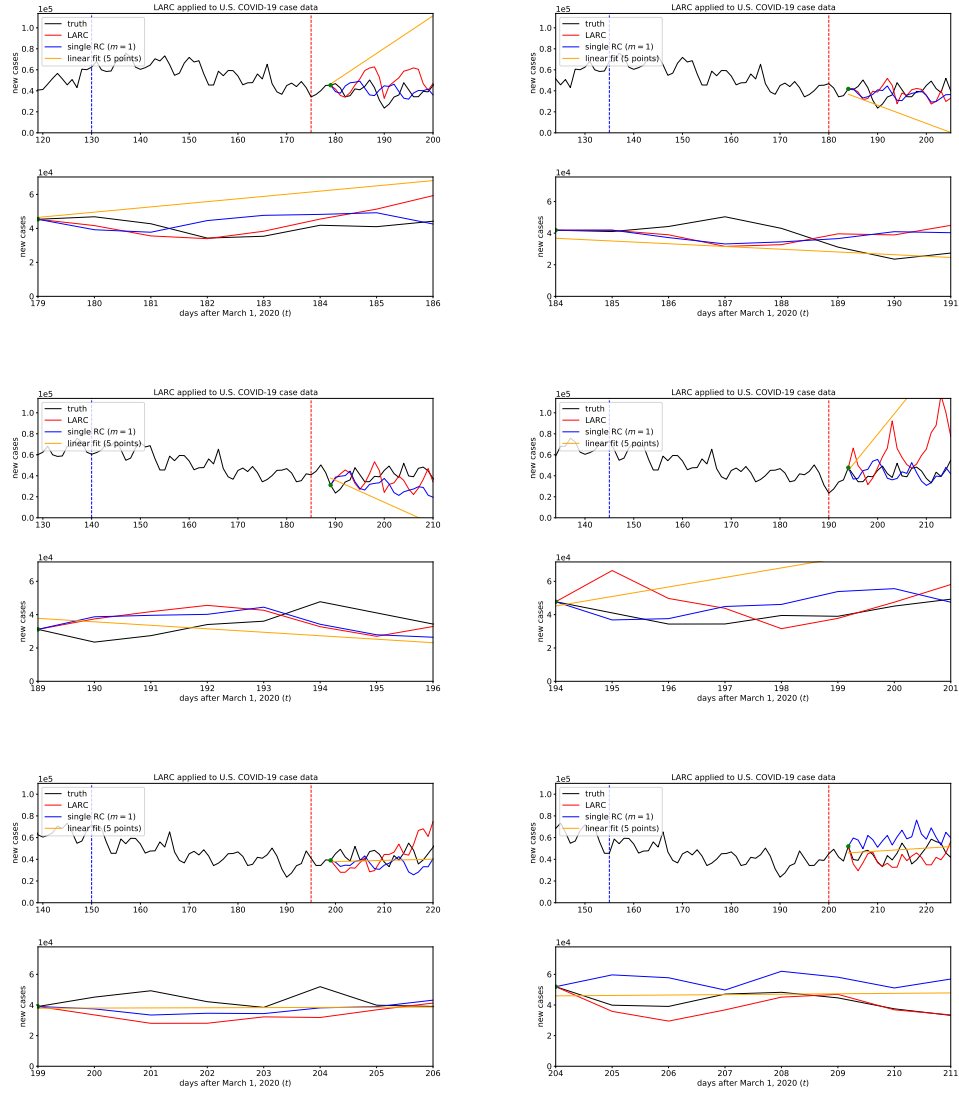


Figure 4.15: Predictions using the library-augmented reservoir computing method (red lines) using five points of training data compared to a single reservoir computer (blue lines) using 21 days of training data and a linear extrapolation (orange lines) using five days of data. Vertical red dashed lines indicate the start of the LARC training data and data used for the linear extrapolation, and vertical blue dashed lines mark the start of the single RC training data. Predictions start on the first day after the green dots. Each bottom panel is a one-week zoom in of the predictions shown in their corresponding top panels. The library used for LARC consists of RCs trained on 50-day overlapping windows in $[30, 125]$.

the New York data set and train an autoencoder with a twelve-dimensional latent space using the first stage of the LARC procedure. As shown in Figure 4.16, there is clear clustering of library members by time.

We are not restricted to constructing a library from a single region. In fact, we may consider constructing a library consisting of RCs trained on data from every state or multiple countries, multiplying the size of the library and increasing the number of examples that a small training signal can be compared to. In Figure 4.17, we demonstrate the latent space embedding of the combined libraries using the U.S. and New York data sets. In this case, not only do we once again observe clustering by time, but there is also clear clustering by region. That is, library members from the U.S. data set are near each other in latent space and library members from the New York data set are near each other in latent space.

We note that clustering is not immediately apparent in every realization of autoencoder training, and introducing more regions makes clustering more difficult, without further increasing the size of the embedding dimension. Furthermore, smaller geographical regions tend to have poorer-quality data (due to smaller sample sizes or irregular reporting) which can introduce examples that negatively impact the overall prediction quality of the library. Such examples can potentially be salvaged by determining a proper transformation, as shown in the Kansas data predictions, or by applying a ‘filter’ that prevents poor examples from being included in the library.

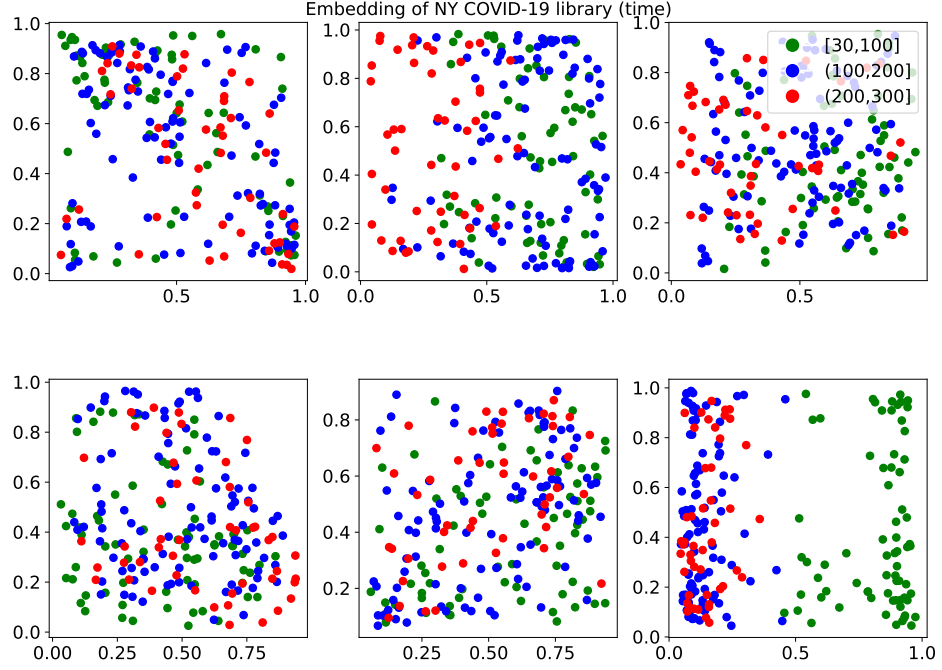


Figure 4.16: Embedding of the New York COVID-19 library into a twelve-dimensional latent space using an autoencoder. The library consists of reservoir computers trained over 50-day overlapping windows of data from $t = 30$ to $t = 300$. Daily case data were transformed using the percent daily change Eq. (4.7) before input into the RCs. Each panel is a projection onto a plane spanned by a pair of coordinate axes. Points are colored according to the first day of the training window, with green points corresponding to starting in $[30, 100]$, blue points corresponding to starting in $(100, 200]$, and red points corresponding to starting in $(200, 300]$.

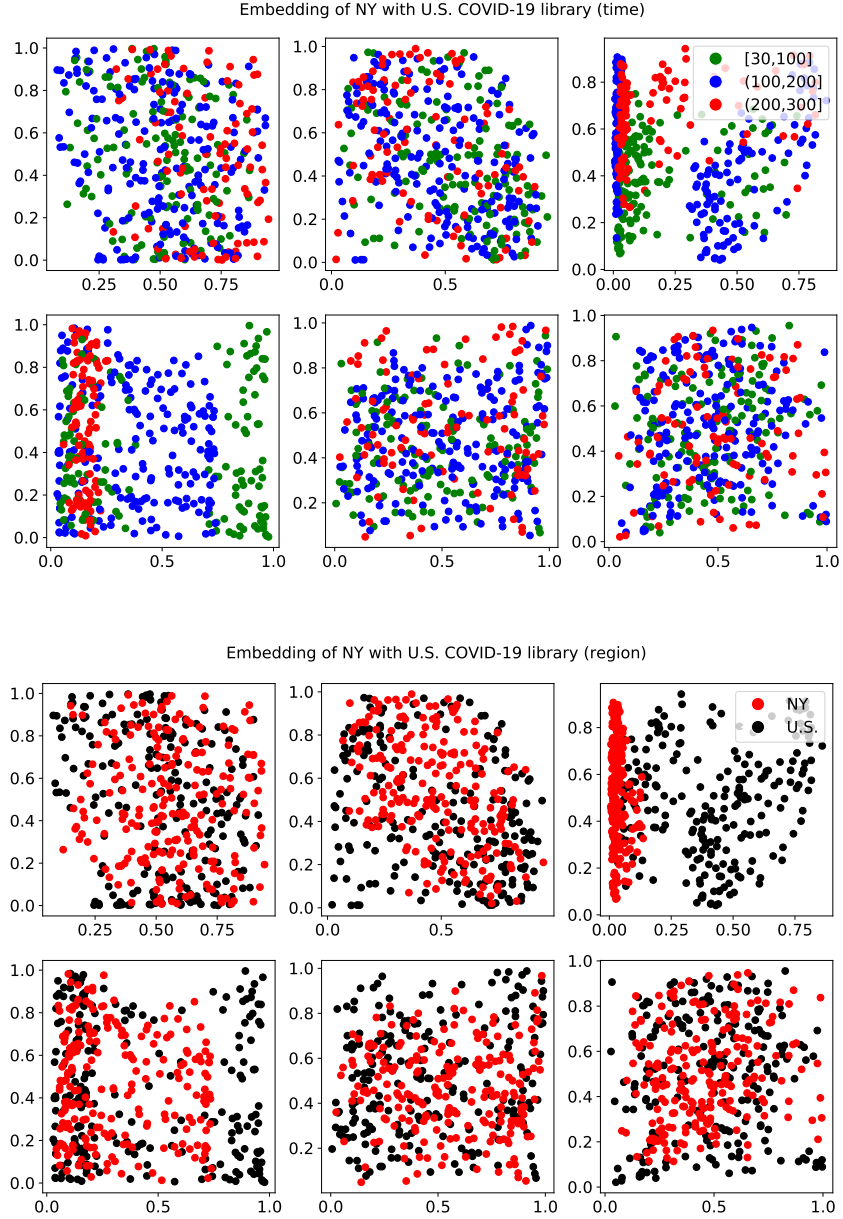


Figure 4.17: Embedding of the combined U.S. and New York COVID-19 library into a twelve-dimensional latent space using an autoencoder. The library consists of reservoir computers trained over 50-day overlapping windows of data from $t = 30$ to $t = 300$. Daily case data were transformed using the percent daily change Eq. (4.7) before input into the RCs. Each panel is a projection onto a plane spanned by a pair of coordinate axes. **Top:** Points are colored according to the first day of the training window, with green points corresponding to starting in $[30, 100]$, blue points corresponding to starting in $(100, 200]$, and red points corresponding to starting in $(200, 300]$. **Bottom:** Points are colored according to region, with red indicating New York and black indicating the U.S.

4.4 Discussion

We have presented an application of reservoir computing to predict U.S. COVID-19 case data at both the national level and individual state levels. Such a task presents multiple challenges, such as low quantities of data and low-quality data. Typical reservoir computing tasks with data generated from a model are able to attain any number of data points (usually thousands) with arbitrary resolution and quality, limited only by the computational resources or artificial limitations imposed by the investigation.

Despite these hurdles, we have demonstrated that a proper choice of smoothing through data preprocessing can salvage poor-quality data, as demonstrated in the Kansas data set. Furthermore, despite access to limited data, we find that 31 data points is sufficient to generate seven-day predictions with reasonable accuracy, occasionally (though not always) within 20% of the true cumulative case number. Unfortunately, a reservoir computing method does not seem to perform significantly better than a simple linear regression estimate that uses fewer data points.

An alternative to a direct reservoir computing approach is the metalearning method of library-augmented reservoir computing. In this method, trained reservoir computers are encoded using an autoencoder, producing a low-dimensional latent space that can be searched to generate a reservoir computer that predicts off of a short training signal. Such an approach has the potential to further overcome the challenge of limited data access by replacing a large number of training points with a large number of library member examples. Ideally, library members would consist of RCs trained on time series exhibiting similar properties to the signals we wish to forecast, but are not required to be from the

exact same process. For example, library members can be trained on different pandemic data than the data we wish to forecast, but underlying properties, such as the spreading rate, may be similar.

It is clear that a direct reservoir computer predictor is not yet able to significantly outperform a simpler approach, such as a linear fit. With the biggest issue being lack of data, we would need to augment our data set, such as through a library-based approach, as discussed, in order to produce more reliable results. Another easy improvement would be to introduce multiple input signals beyond just the case data, such as death counts, hospitalization numbers, or mobility data. While increasing the number of inputs provides access to more data, we do note that each additional input would also need to be predicted by the reservoir computer.

Other than supplementing with additional data, we can consider augmenting the basic reservoir computing architecture. For instant, we can introduce a network of reservoir computers connected to represent mobility between different subregions (i.e., states). In a similar manner, the basic reservoir computer can be replaced with a hybrid system, in which data is input in parallel with model predictions, such as from a basic SIR model or any successful variation already applied to COVID-19 data. While no longer truly ‘model-free, the model is able to guide the reservoir computer predictions, despite potential inaccuracies in the model itself.

Appendix A: Supplementary material for Chapter 2

A.1 Locally treelike features

Here we demonstrate the locally treelike features for our simulated networks as they are grown through evolutionary time. In the semi-annealed approximation described in the main text, which allows us to map the stability of synchronous Boolean dynamics to an appropriately defined percolation problem, we make the assumption that pairs of nodes are rarely connected by multiple short paths of the same length (in the same direction). This is a weaker version of the more widely used locally treelike assumption. For our simulated networks, we consider all pairs of nodes connected by a path of length two and calculate the fraction that are connected by at least one other path of length two (in the same direction). This bi-parallel motif (i.e., $A \rightarrow B \rightarrow C$ **and** $A \rightarrow D \rightarrow C$) is the smallest network motif that is relevant for the case of synchronous update of nodes in a Boolean network. The smaller feedforward motif (i.e., $A \rightarrow B \rightarrow C$ **and** $A \rightarrow C$) does not cause an issue when applying the semi-annealed approximation because signal information from A to C that goes through B does not arrive in the same time step as the information from that goes directly from A to C . Results show in the figure below are for a network of size $N = 10^5$. We see that the prevalence of the bi-parallel motif is very small for full duration of the link addition processes considered in the main text.

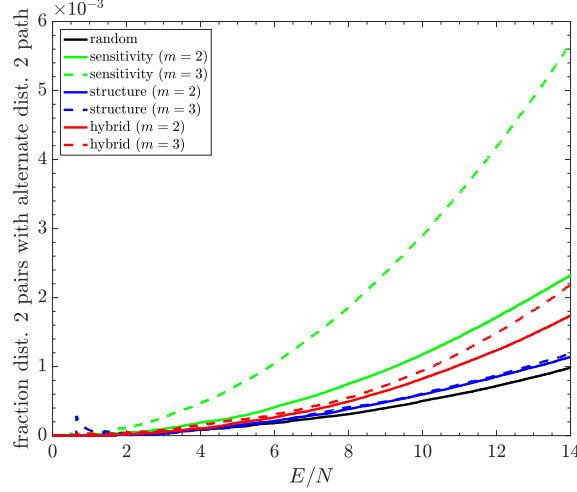


Figure A.1: The fractions of nodes connected by a directed path of length two connected by at least one other directed path of length two.

A.2 Selection with $m = N$

In this section, we consider the structure-based rule that relies on full global information at each time step, i.e., $m = N$, with ties broken randomly. Recall that N is the number of genes and m is the number of candidate genes considered for both the source and target node each time a new link is added to the system. Results are shown below for a network of size $N = 1000$, averaged over 100 realizations. Due to the significant increase in runtime for $m = N$ compared with $m = 2$ or $m = 3$ reported in the main text, we used a smaller value of N for this study. Even at this smaller value of N we can see that selection with $m = N$ significantly outperforms all others in suppressing the growth of connected components.

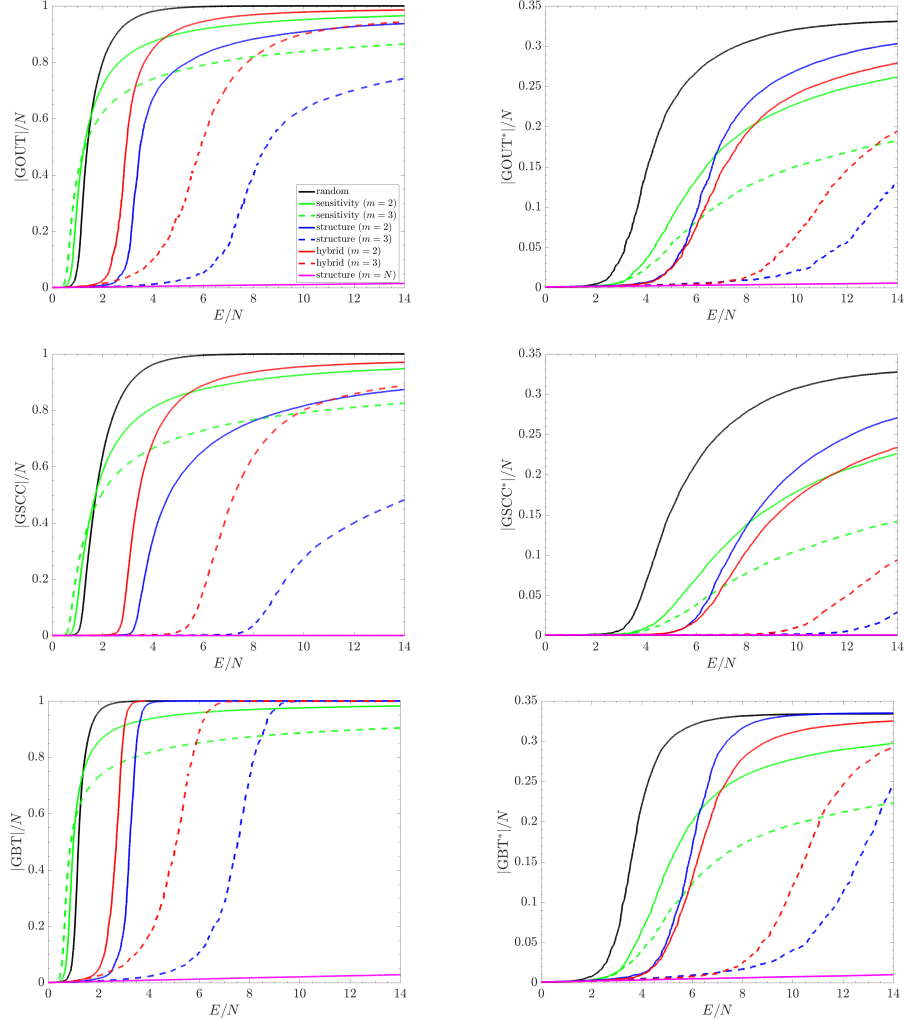


Figure A.2: Growth of the connected components on the full network (left) and the damaged subnetwork (right). Results shown are the averages of 100 runs on networks of size $N = 1000$. We omit GIN and GIN* due to their symmetry with GOUT and GOUT*, respectively.

Appendix B: Complications with data preprocessing of COVID-19 case data

In this appendix, we consider several issues that need addressing when developing a proper transformation for the COVID-19 case data.

B.1 Negative number of cases reported

To account for overestimates in the number of cases, several regions have single days with a “negative” number of cases. Without taking into account reporting errors, the assumption of the underlying model represented by the reservoir computer would require the number of daily cases to always be non-negative. For simplicity, we focus our results in the main text only on regions and times in which all cases reported are non-negative. An alternative, simple solution would be to zero-out all days with negative cases. In most cases, the number of negative cases reported is small, so that the overall totals are minimally impacted. For more extreme cases, one could equally divide the negative cases across all previous days of data.

B.2 Irregular reporting

While most regions report daily cases, certain locations report the number of new cases accumulated every few (e.g. three) days. For transformations in the form

$$u(t) = \frac{\hat{x}'(t)}{\hat{x}(t)}, \quad (\text{B.1})$$

several days of zero cases followed by a single day with a large non-zero number of cases will cause a “spike” in $u(t)$ that becomes challenging to predict. A form of smoothing or averaging will be required to avoid spikes in data, dividing their contribution over several days.

Even in less extreme examples, where cases are reported every day, there is a clear oscillation in reporting frequency that follows the days of the week. These well-behaved oscillations are potentially useful in generating realistic-looking predictions that also match these oscillations. We have attempted to incorporate the 7-day oscillation period into the reservoir computer input, but such methods have shown negligible improvement in prediction quality.

B.3 Zero cases reported

Unlike negative number of cases, a day with zero cases is valid within the framework of the underlying model. However, the transformation (B.1) is invalid for a denominator of zero. Smoothing has the potential to prevent this issue, but fails when a trail of zeros is longer than the smoothing window. As a straightforward solution, we replace

$x(t)$ with $\tilde{x}(t) = x(t) + 1$. In general, the data can be shifted by any value c , which introduces an additional hyperparameter to the system. During phases of epidemic spread, the number of daily cases are on the order of hundreds (at the state level) to several thousands (at the national level). Thus, shifting all values by one will have negligible impact on the total case counts over this period. Furthermore, after training the reservoir computer to predict $\tilde{x}(t)$, we can easily recover $x(t)$ by shifting back. Note that in cases where $\hat{x}'(t)$ is calculated by differencing or the slope of a linear fit, the vertical shift will effectively cancel out. Our main concern is with guaranteeing $\hat{x}(t)$ never becomes zero.

B.4 Invertibility

Even if a transformation of the data is well-defined, we run into issues if the inverse has a limited domain. Specifically, restrictions on $u(t)$ resulting from the forward transformation may not be preserved when future values are generated from a reservoir computer. For instance, the transformation

$$u(t) = \tanh \left(\frac{\tilde{x}(t) - \tilde{x}(t-1)}{\tilde{x}(t-1)} \right), \quad (\text{B.2})$$

has shown to be effective in producing predictions in initial tests. The main feature of the hyperbolic tangent is to suppress large intermittent spikes. However, the inverse transformation requires a domain of $(-1, 1)$, which is not enforced by the reservoir computer. In the case of a prediction extending beyond this domain, the prediction is lost from lack of invertibility. Even worse, all future predicted values are lost, since each subsequent prediction depends multiplicatively on the value of the last. In the specific case of the

hyperbolic tangent, values that exist near 1 (even without passing it) can also lead to blow up of the inverse transformation, which is similarly detrimental, despite the inverse transformation existing. Consequently, it is prudent to restrict ourselves to transformations $u(t)$ that have well-defined inverses for any possible predicted value.

Bibliography

- [1] B. J. Liebeskind, C. D. McWhite, and E. M. Marcotte, “Towards consensus gene ages,” *Genome Biol Evol*, vol. 8, no. 6, pp. 1812–1823, 2016. (doi: 10.1093/gbe/evw113).
- [2] B. Alexander, A. Pushkar, and M. Girvan, “Phase transitions and assortativity in models of gene regulatory networks evolved under different selection processes,” *J R Soc Interface*, vol. 18, Apr. 2021. (doi: 10.1098/rsif.2020.0790).
- [3] S. Kauffman, “Metabolic stability and epigenesis in randomly constructed genetic nets,” *J Theor Biol*, vol. 22, no. 3, pp. 437 – 467, 1969. (doi: 10.1016/0022-5193(69)90015-0).
- [4] A. Pomerance, E. Ott, M. Girvan, and W. Losert, “The effect of network topology on the stability of discrete state models of genetic control,” *P Natl Acad Sci USA*, vol. 106, no. 20, pp. 8209–8214, 2009. (doi: 10.1073/pnas.0900142106).
- [5] S. Squires, E. Ott, and M. Girvan, “Dynamical instability in boolean networks as a percolation problem,” *Phys Rev Lett*, vol. 109, p. 085701, Aug 2012. (doi: 10.1103/PhysRevLett.109.085701).
- [6] D. Achlioptas, R. M. D’Souza, and J. Spencer, “Explosive percolation in random networks,” *Science*, vol. 323, no. 5920, pp. 1453–1455, 2009. (doi: 10.1126/science.1167782).
- [7] A. Chicoli and D. A. Paley, “Probabilistic information transmission in a network of coupled oscillators reveals speed-accuracy trade-off in responding to threats,” *Chaos*, vol. 26, no. 11, p. 116311, 2016. (doi: 10.1063/1.4966682).
- [8] R. Solé and S. Valverde, “Evolving complexity: how tinkering shapes cells, software and ecological networks,” *Philos T R Soc B*, vol. 375, no. 1796, p. 20190325, 2020. (doi: 10.1098/rstb.2019.0325).
- [9] T. Mora and W. Bialek, “Are biological systems poised at criticality?,” *J Stat Phys*, vol. 144, pp. 268–302, Jul 2011. (doi: 10.1007/s10955-011-0229-4).
- [10] B. C. Daniels, H. Kim, D. Moore, S. Zhou, H. B. Smith, B. Karas, S. A. Kauffman, and S. I. Walker, “Criticality distinguishes the ensemble of biological regulatory

- networks,” *Phys Rev Lett*, vol. 121, p. 138102, Sep 2018. (doi: 10.1103/PhysRevLett.121.138102).
- [11] E. Martínez, M. Villani, L. La Rocca, S. A. Kauffman, and R. Serra, “Dynamical criticality in gene regulatory networks,” *Complexity*, vol. 2018, p. 5980636, 2018. (doi: 10.1155/2018/5980636).
 - [12] B. Derrida and Y. Pomeau, “Random networks of automata: A simple annealed approximation,” *Europhys Lett*, vol. 1, no. 2, p. 45, 1986. (doi: 10.1209/0295-5075/1/2/001).
 - [13] M. Aldana and P. Cluzel, “A natural class of robust networks,” *P Natl Acad Sci USA*, vol. 100, no. 15, pp. 8710–8714, 2003. (doi: 10.1073/pnas.1536783100).
 - [14] S. Squires, K. Sytzu, D. Alcalá, T. M. Antonsen, E. Ott, and M. Girvan, “Weakly explosive percolation in directed networks,” *Phys Rev E*, vol. 87, p. 052127, May 2013. (doi: 10.1103/PhysRevE.87.052127).
 - [15] S. Kauffman, C. Peterson, B. Samuelsson, and C. Troein, “Genetic networks with canalizing boolean rules are always stable,” *P Natl Acad Sci USA*, vol. 101, no. 49, pp. 17102–17107, 2004. (doi: 10.1073/pnas.0407783101).
 - [16] A. Pomerance, M. Girvan, and E. Ott, “Stability of boolean networks with generalized canalizing rules,” *Phys Rev E*, vol. 85, no. 4, p. 046106, 2012. (doi: 10.1103/PhysRevE.85.046106).
 - [17] C. Seshadhri, A. M. Smith, Y. Vorobeychik, J. R. Mayo, and R. C. Armstrong, “Characterizing short-term stability for boolean networks over any distribution of transfer functions,” *Phys Rev E*, vol. 94, no. 1, p. 012301, 2016. (doi: 10.1103/PhysRevE.94.012301).
 - [18] S. Squires, A. Pomerance, M. Girvan, and E. Ott, “Stability of boolean networks: The joint effects of topology and update rules,” *Phys Rev E*, vol. 90, no. 2, p. 022814, 2014. (doi: 10.1103/PhysRevE.90.022814).
 - [19] S. Bornholdt and T. Rohlf, “Topological evolution of dynamical networks: Global criticality from local dynamics,” *Phys Rev Lett*, vol. 84, no. 26, p. 6114, 2000. (doi: 10.1103/PhysRevLett.84.6114).
 - [20] P. Oikonomou and P. Cluzel, “Effects of topology on network evolution,” *Nat Phys*, vol. 2, no. 8, p. 532, 2006. (doi: 10.1038/nphys359).
 - [21] M. Aldana, E. Balleza, S. Kauffman, and O. Resendiz, “Robustness and evolvability in genetic regulatory networks,” *J Theor Biol*, vol. 245, no. 3, pp. 433–448, 2007. (doi: 10.1016/j.jtbi.2006.10.027).
 - [22] P. Grassberger, C. Christensen, G. Bizhani, S.-W. Son, and M. Paczuski, “Explosive percolation is continuous, but with unusual finite size behavior,” *Phys Rev Lett*, vol. 106, no. 22, p. 225701, 2011. (doi: 10.1103/PhysRevLett.106.225701).

- [23] O. Riordan and L. Warnke, “Explosive percolation is continuous,” *Science*, vol. 333, no. 6040, pp. 322–324, 2011. (doi: 10.1126/science.1206241).
- [24] R. M. D’Souza and J. Nagler, “Anomalous critical and supercritical phenomena in explosive percolation,” *Nat Phys*, vol. 11, no. 7, p. 531, 2015. (doi: 10.1038/nphys3378).
- [25] W. Viles, C. E. Ginestet, A. Tang, M. A. Kramer, and E. D. Kolaczyk, “Percolation under noise: Detecting explosive percolation using the second-largest component,” *Phys Rev E*, vol. 93, no. 5, p. 052301, 2016. (doi: 10.1103/PhysRevE.93.052301).
- [26] A. Waagen, R. M. D’Souza, and T.-C. Lu, “Explosive percolation on directed networks due to monotonic flow of activity,” *Phys Rev E*, vol. 96, no. 1, p. 012317, 2017. (doi: 10.1103/PhysRevE.96.012317).
- [27] R. M. D’Souza, J. Gómez-Gardeñes, J. Nagler, and A. Arenas, “Explosive phenomena in complex networks,” *Adv Phys*, vol. 68, no. 3, pp. 123–223, 2019. (doi: 10.1080/00018732.2019.1650450).
- [28] J. M. Carlson and J. Doyle, “Highly optimized tolerance: A mechanism for power laws in designed systems,” *Phys Rev E*, vol. 60, pp. 1412–1427, Aug 1999. (doi: 10.1103/PhysRevE.60.1412).
- [29] J. M. Carlson and J. Doyle, “Highly optimized tolerance: Robustness and design in complex systems,” *Phys Rev Lett*, vol. 84, pp. 2529–2532, Mar 2000. (doi: 10.1103/PhysRevLett.84.2529).
- [30] V. van Noort, B. Snel, and M. A. Huynen, “Predicting gene function by conserved co-expression,” *Trends Genet*, vol. 19, no. 5, pp. 238–242, 2003. (doi: 10.1016/S0168-9525(03)00056-8).
- [31] S. A. Teichmann and M. M. Babu, “Gene regulatory network growth by duplication,” *Nat Genet*, vol. 36, no. 5, p. 492, 2004. (doi: 10.1038/ng1340).
- [32] P. D. Kuo, W. Banzhaf, and A. Leier, “Network topology and the evolution of dynamics in an artificial genetic regulatory network model created by whole genome duplication and divergence,” *Biosystems*, vol. 85, no. 3, pp. 177–200, 2006. (doi: 10.1016/j.biosystems.2006.01.004).
- [33] S. Manrubia and J. A. Cuesta, “Evolution on neutral networks accelerates the ticking rate of the molecular clock,” *J R Soc Interface*, vol. 12, no. 102, p. 20141010, 2015. (doi: 10.1098/rsif.2014.1010).
- [34] L. Demetrius and T. Manke, “Robustness and network evolution—an entropic principle,” *Physica A*, vol. 346, pp. 682–696, Feb. 2005. (doi: 10.1016/j.physa.2004.07.011).

- [35] J. Demongeot, A. Elena, and S. Sené, “Robustness in regulatory networks: A multi-disciplinary approach,” *Acta Biotheor*, vol. 56, no. 1, pp. 27–49, 2008. (doi: 10.1007/s10441-008-9029-x).
- [36] S. Melnik, A. Hackett, M. A. Porter, P. J. Mucha, and J. P. Gleeson, “The unreasonable effectiveness of tree-based theory for networks with clustering,” *Phys Rev E*, vol. 83, p. 036112, Mar 2011. (doi: 10.1103/PhysRevE.83.036112).
- [37] S. Chandra, E. Ott, and M. Girvan, “Critical network cascades with re-excitable nodes: Why treelike approximations usually work, when they break down, and how to correct them,” *Phys Rev E*, vol. 101, p. 062304, Jun 2020. (doi: 10.1103/PhysRevE.101.062304).
- [38] M. E. J. Newman, S. H. Strogatz, and D. J. Watts, “Random graphs with arbitrary degree distributions and their applications,” *Phys Rev E*, vol. 64, p. 026118, Jul 2001. (doi: 10.1103/PhysRevE.64.026118).
- [39] P. Erdős and A. Rényi, “On the evolution of random graphs,” *Publ Math Inst Hung Acad Sci*, vol. 5, pp. 17–61, 1960.
- [40] R. A. da Costa, S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes, “Explosive percolation transition is actually continuous,” *Phys Rev Lett*, vol. 105, no. 25, p. 255701, 2010. (doi: 10.1103/PhysRevLett.105.255701).
- [41] A. Santos-Zavaleta, H. Salgado, S. Gama-Castro, M. Sánchez-Pérez, L. Gómez-Romero, D. Ledezma-Tejeida, J. S. García-Sotelo, K. Alquicira-Hernández, L. J. Muñoz-Rascado, P. Peña-Loredo, C. Ishida-Gutiérrez, D. A. Velázquez-Ramírez, V. Del Moral-Chávez, C. Bonavides-Martínez, C.-F. Méndez-Cruz, J. Galagan, and J. Collado-Vides, “RegulonDB v 10.5: tackling challenges to unify classic and high throughput knowledge of gene regulation in *E. coli* K-12,” *Nucleic Acids Res*, vol. 47, pp. D212–D220, 11 2018. (doi: 10.1093/nar/gky1077).
- [42] C. Chen, D. Zhang, T. R. Hazbun, and M. Zhang, “Inferring gene regulatory networks from a population of yeast segregants,” *Sci Rep*, vol. 9, no. 1, p. 1197, 2019. (doi: 10.1038/s41598-018-37667-4).
- [43] D. Y. Rhee, D.-Y. Cho, B. Zhai, M. Slattery, L. Ma, J. Mintseris, C. Y. Wong, K. P. White, S. E. Celniker, T. M. Przytycka, S. P. Gygi, R. A. Obar, and S. Artavanis-Tsakonas, “Transcription factor networks in *drosophila melanogaster*,” *Cell Rep*, vol. 8, no. 6, pp. 2031–2043, 2014. (doi: 10.1016/j.celrep.2014.08.038).
- [44] H. Han, J.-W. Cho, S. Lee, A. Yun, H. Kim, D. Bae, S. Yang, C. Y. Kim, M. Lee, E. Kim, S. Lee, B. Kang, D. Jeong, Y. Kim, H.-N. Jeon, H. Jung, S. Nam, M. Chung, J.-H. Kim, and I. Lee, “TRRUST v2: an expanded reference database of human and mouse transcriptional regulatory interactions,” *Nucleic Acids Res*, vol. 46, pp. D380–D386, 10 2017. (doi: 10.1093/nar/gkx1013).

- [45] M. B. Gerstein, A. Kundaje, M. Hariharan, S. G. Landt, K.-K. Yan, C. Cheng, X. J. Mu, E. Khurana, J. Rozowsky, R. Alexander, *et al.*, “Architecture of the human regulatory network derived from encode data,” *Nature*, vol. 489, no. 7414, p. 91, 2012. (doi: 10.1038/nature11245).
- [46] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, “Novel type of phase transition in a system of self-driven particles,” *Phys Rev Lett*, vol. 75, pp. 1226–1229, Aug 1995. (doi: 10.1103/PhysRevLett.75.1226).
- [47] K. P. O’Keefe, H. Hong, and S. H. Strogatz, “Oscillators that sync and swarm,” *Nat Commun*, vol. 8, p. 1504, 2017. (doi: 10.1038/s41467-017-01190-3).
- [48] D. Levis, A. Diaz-Guilera, I. Pagonabarraga, and M. Starnini, “Flocking-enhanced social contagion,” *Phys Rev Res*, vol. 2, p. 032056, Sep 2020. (doi: 10.1103/PhysRevResearch.2.032056).
- [49] M. E. J. Newman, “Spread of epidemic disease on networks,” *Phys Rev E*, vol. 66, July 2002. (doi: 10.1103/physreve.66.016128).
- [50] J. R. C. Piqueira, B. F. Navarro, and L. H. A. Monteiro, “Epidemiological models applied to viruses in computer networks,” *J Comput Sci*, vol. 1, pp. 31–34, Jan. 2005. (doi: 10.3844/jcssp.2005.31.34).
- [51] F. Jin, E. Dougherty, P. Saraf, Y. Cao, and N. Ramakrishnan, “Epidemiological modeling of news and rumors on twitter,” in *Proceedings of the 7th Workshop on Social Network Mining and Analysis - SNAKDD '13*, ACM Press, 2013. (doi: 10.1145/2501025.2501027).
- [52] K. Zhu and L. Ying, “Information source detection in the SIR model: A sample-path-based approach,” *IEEE ACM T Network*, vol. 24, pp. 408–421, Feb. 2016. (doi: 10.1109/tnet.2014.2364972).
- [53] D. Sumpter, *Collective animal behavior*. Princeton, N.J: Princeton University Press, 2010.
- [54] I. Aihara, H. Kitahata, K. Yoshikawa, and K. Aihara, “Mathematical modeling of frogs’ calling behavior and its possible application to artificial life and robotics,” *Artif Life Robotics*, vol. 12, pp. 29–32, Mar. 2008. (doi: 10.1007/s10015-007-0436-x).
- [55] D. Helbing, I. Farkas, and T. Vicsek, “Simulating dynamical features of escape panic,” *Nature*, vol. 407, pp. 487–490, Sept. 2000. (doi: 10.1038/35035023).
- [56] D. J. Low, “Following the crowd,” *Nature*, vol. 407, pp. 465–466, Sept. 2000. (doi: 10.1038/35035192).
- [57] A. E. Motter, S. A. Myers, M. Anghel, and T. Nishikawa, “Spontaneous synchrony in power-grid networks,” *Nat Phys*, vol. 9, pp. 191–197, Feb. 2013. (doi: 10.1038/nphys2535).

- [58] Y. Kuramoto, “Self-entrainment of a population of coupled non-linear oscillators,” in *International Symposium on Mathematical Problems in Theoretical Physics*, pp. 420–422, Springer-Verlag, 1974. (doi: 10.1007/bfb0013365).
- [59] Y. Kuramoto, *Chemical Oscillations, Waves, and Turbulence*. Springer Berlin Heidelberg, 1984. (doi: 10.1007/978-3-642-69689-3).
- [60] A. Pluchino, V. Latora, and A. Rapisarda, “Compromise and synchronization in opinion dynamics,” *Eur Phys J B*, vol. 50, pp. 169–176, Mar. 2006. (doi: 10.1140/epjb/e2006-00131-0).
- [61] H. Hong and S. H. Strogatz, “Conformists and contrarians in a kuramoto model with identical natural frequencies,” *Phys Rev E*, vol. 84, Oct. 2011. (doi: 10.1103/phys-reve.84.046202).
- [62] K. Kaneko, “Globally coupled circle maps,” *Physica D*, vol. 54, no. 1, pp. 5–19, 1991. (doi: 10.1016/0167-2789(91)90103-G).
- [63] G. Barlev, M. Girvan, and E. Ott, “Map model for synchronization of systems of many coupled oscillators,” *Chaos*, vol. 20, no. 2, p. 023109, 2010. (doi: 10.1063/1.3357983).
- [64] W. O. Kermack, A. G. McKendrick, and G. T. Walker, “A contribution to the mathematical theory of epidemics,” *P R Soc A*, vol. 115, no. 772, pp. 700–721, 1927. (doi: 10.1098/rspa.1927.0118).
- [65] L. J. Allen, “Some discrete-time SI, SIR, and SIS epidemic models,” *Math Biosci*, vol. 124, no. 1, pp. 83–105, 1994. (doi: 10.1016/0025-5564(94)90025-6).
- [66] J. Kephart and S. White, “Directed-graph epidemiological models of computer viruses,” in *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 343–359, IEEE Comput. Soc. Press, 1991. (doi: 10.1109/risp.1991.130801).
- [67] D. J. Daley and D. G. Kendall, “Epidemics and rumours,” *Nature*, vol. 204, pp. 1118–1118, Dec. 1964. (doi: 10.1038/2041118a0).
- [68] H. Daido, “Discrete-Time Population Dynamics of Interacting Self-Oscillators,” *Prog Theor Phys*, vol. 75, pp. 1460–1463, 06 1986. (doi: 10.1143/PTP.75.1460).
- [69] J. Arino and S. Portet, “A simple model for COVID-19,” *Infec Dis Modelling*, vol. 5, pp. 309–315, 2020. (doi: 10.1016/j.idm.2020.04.002).
- [70] I. Korolev, “Identification and estimation of the SEIRD epidemic model for COVID-19,” *J Econometrics*, vol. 220, pp. 63–85, Jan. 2021. (doi: 10.1016/j.jeconom.2020.07.038).
- [71] A. Karaivanov, “A social network model of COVID-19,” *PLOS ONE*, vol. 15, p. e0240878, Oct. 2020. (doi: 10.1371/journal.pone.0240878).

- [72] K. jae Kim, “Financial time series forecasting using support vector machines,” *Neurocomputing*, vol. 55, no. 1, pp. 307–319, 2003. (doi: 10.1016/S0925-2312(03)00372-2).
- [73] H. Yan and H. Ouyang, “Financial time series prediction based on deep learning,” *Wireless Pers Commun*, vol. 102, pp. 683–700, Dec. 2017. (doi: 10.1007/s11277-017-5086-2).
- [74] Z. Han, J. Zhao, H. Leung, K. F. Ma, and W. Wang, “A review of deep learning models for time series prediction,” *IEEE Sens J*, vol. 21, pp. 7833–7848, Mar. 2021. (doi: 10.1109/jsen.2019.2923982).
- [75] M. Han, J. Xi, S. Xu, and F.-L. Yin, “Prediction of chaotic time series based on the recurrent predictor neural network,” *IEEE T Signal Proces*, vol. 52, pp. 3409–3416, Dec. 2004. (doi: 10.1109/tsp.2004.837418).
- [76] S. Pappas, L. Ekonomou, D. Karamousantas, G. Chatzarakis, S. Katsikas, and P. Liatsis, “Electricity demand loads modeling using AutoRegressive Moving Average (ARMA) models,” *Energy*, vol. 33, no. 9, pp. 1353–1360, 2008. (doi: 10.1016/j.energy.2008.05.008).
- [77] D. Ömer Faruk, “A hybrid neural network and ARIMA model for water quality time series prediction,” *Eng App Artif Intel*, vol. 23, no. 4, pp. 586–594, 2010. (doi: 10.1016/j.engappai.2009.09.015).
- [78] S. H. Holan, R. Lund, and G. Davis, “The ARMA alphabet soup: A tour of ARMA model variants,” *Stat Surveys*, vol. 4, no. none, pp. 232 – 274, 2010. (doi: 10.1214/09-SS060).
- [79] R. Gupta, G. Pandey, P. Chaudhary, and S. Pal, “SEIR and regression model based COVID-19 outbreak predictions in india,” *medRxiv*, Apr. 2020. Preprint. (doi: 10.1101/2020.04.01.20049825).
- [80] B. F. Maier and D. Brockmann, “Effective containment explains subexponential growth in recent confirmed COVID-19 cases in china,” *Science*, vol. 368, pp. 742–746, Apr. 2020. (doi: 10.1126/science.abb4557).
- [81] R. Gupta and S. K. Pal, “Trend analysis and forecasting of COVID-19 outbreak in india,” *medRxiv*, Mar. 2020. Preprint. (doi: 10.1101/2020.03.26.20044511).
- [82] Z. Hu, Q. Ge, S. Li, L. Jin, and M. Xiong, “Artificial intelligence forecasting of covid-19 in china,” 2020. Preprint.
- [83] Z. Yang, Z. Zeng, K. Wang, S.-S. Wong, W. Liang, M. Zanin, P. Liu, X. Cao, Z. Gao, Z. Mai, J. Liang, X. Liu, S. Li, Y. Li, F. Ye, W. Guan, Y. Yang, F. Li, S. Luo, Y. Xie, B. Liu, Z. Wang, S. Zhang, Y. Wang, N. Zhong, and J. He, “Modified SEIR and AI prediction of the epidemics trend of COVID-19 in china under public health interventions,” *J Thorac Dis*, vol. 12, pp. 165–174, Mar. 2020. (doi: 10.21037/jtd.2020.02.64).

- [84] A. Tomar and N. Gupta, “Prediction for the spread of COVID-19 in india and effectiveness of preventive measures,” *Sci Total Environ*, vol. 728, p. 138762, Aug. 2020. (doi: 10.1016/j.scitotenv.2020.138762).
- [85] M. Lukoševičius, “A practical guide to applying echo state networks,” in *Lecture Notes in Computer Science*, pp. 659–686, Springer Berlin Heidelberg, 2012. (doi: 10.1007/978-3-642-35289-8_36).
- [86] Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, and E. Ott, “Reservoir observers: Model-free inference of unmeasured variables in chaotic systems,” *Chaos*, vol. 27, p. 041102, Apr. 2017. (doi: 10.1063/1.4979665).
- [87] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, “Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach,” *Phys Rev Lett*, vol. 120, Jan. 2018. (doi: 10.1103/physrevlett.120.024102).
- [88] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, “Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data,” *Chaos*, vol. 27, p. 121102, Dec. 2017. (doi: 10.1063/1.5010300).
- [89] D. Patel, D. Canaday, M. Girvan, A. Pomerance, and E. Ott, “Using machine learning to predict statistical properties of non-stationary dynamical processes: System climate, regime transitions, and the effect of stochasticity,” *Chaos*, vol. 31, p. 033149, Mar. 2021. (doi: 10.1063/5.0042598).
- [90] J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott, “Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model,” *Chaos*, vol. 28, p. 041101, Apr. 2018. (doi: 10.1063/1.5028373).
- [91] D. Canaday and A. Pomerance, “Library-augmented reservoir computing: A meta-learning approach to time series prediction with limited data.” Preprint.
- [92] D. C. Farrow, L. C. Brooks, A. Rumack, R. J. Tibshirani, and R. Rosenfeld, “Delphi epidata API,” 2015. <https://github.com/cmu-delphi/delphi-epidata>.